



IST-2001-34091 SCOUT

D4.3.1

Description and Specification of the SCOUT hardware validators and SCOUT Middleware Demonstrator

Contractual Date of Delivery to the CEC:	31.01.2003
Actual Date of Delivery to the CEC:	11.02.2003
Author(s):	Philippe du Reau, David Redmill, Eiman Mohyeldin, Zoran Golubicic, Robert Hirschfeld, Michael Fahrmaier, Chris Salzmair, Peter Dornbusch
Participant(s):	SM, FTR& D, UoB, DCLE, TTI
Workpackage:	WP4.3: Validation and Performance Assessment
Est. person months:	24
Security:	PU
Nature:	R
Version:	1.2
Total number of pages:	42

Abstract:

This document describes the specifications of the demonstrations to be performed in the project.

1/ Reconfigurable RF TX chain

The proposed RF TX demonstrator hardware is based on commercially available components but at the same time it is near to the solution obtained during the analysis of the RF part of Scout terminal.

2/ Link adaptation on hardware validator

The document provides the specifications of the demonstration to be carried out with the hardware validator: link adaptation and video demonstration. The link adaptation functionalities to be demonstrated in the physical layer are defined in a precise manner. Then the conditions required for installing the video demonstration are described. Finally the stages and expected benefits of the experimentation are briefly shown.

3/ Video demonstration

The interaction between the video systems that are developed and the hardware validator is described. The aim is to provide a simple convenient and flexible interface which will allow the video systems to be demonstrated over the hardware validator, and provide meaningful results which are representative of those anticipated from a complete system.

4/ Middleware demonstration

The general objectives are to adequately demonstrate a middleware concept for reconfiguration. The middleware demonstrator should support several tasks and requirements. The middleware demonstrator emulates the mobile channel according to the variation of the network bandwidths defined in WLAN, UMTS, GPRS and the delay experienced by the application. A web browsing application will be demonstrated.

Keyword list: Middleware, WLAN, web service definition language, video codec, trading service negotiation, reconfiguration, shadow channel, linearisation, component sharing, up and down conversion, specification, demonstration, experimentation, real time reconfiguration, slot by slot reconfiguration, link adaptation, adaptive modulation, reconfigurable error correction code, BPSK-QPSK-16QAM modulations, convolutional code, scalable video, wideband CDMA, UTRA, transmission quality parameter, propagation channel, signal to noise ratio, bit error rate, link budget.

Executive Summary

This deliverable describes the specifications of the demonstrations to be performed in WP4.3:

- in the technology level (A4.3.3): reconfigurable RF transmitter chain (section 2);
- in the physical layer (A4.3.3): real time reconfigurability with a hardware validator (section 3);
- in the application level (A4.3.4, A4.3.5): a scalable video demonstration on the hardware validator (section 4);
- in the middleware domain (A4.3.6): demonstration of a middleware concept for reconfiguration (section 5).

1/ Technology for reconfigurable RF Transmit chain

Most important thing in this study is a way of the including shadow channel in RF Tx chain. IMD in power amplifier during transition period is avoided because two signals are not amplified with same amplifier. Instead, reconfiguration of balanced power amplifier is used and separate amplifiers are used for particular channels. Our realization doesn't need additional amplifier and additional power consumption and additional back off (maximum efficiency). This is full advantage of terminal reconfiguration architecture.

Using bank of microwave LO signals (instead of synthesized PLL) speed of standard reconfiguration can be significantly below duration of shortest symbol. At the same time phase noise of microwave LO is optimized because all LO signals are derived from phase locked fixed frequency oscillator (with high Q resonator and best phase noise at higher frequency offsets where phase noise of multiplied reference is not usable).

Frequency changing inside one standard is performed by using DDS circuit. In this way the phase continuity during transition period is provided. BB signal is generated by PC host supplied with specialized data acquisition AD and DA cards. These also enable monitoring of transmitter parameters and calculating additional predistortion for output linearisation.

2/ Real time reconfigurability with hardware validator

In TRUST project the hardware validator has been used to unveil some benefits of the adaptive modulation. In SCOUT project it will be used for two demonstrations:

- a real time reconfiguration of the radio interface,
- a service demonstration which will be installed onto the previous reconfiguration scenario.

The real time reconfiguration is a link adaptation to the instantaneous fluctuations of the propagation channel. The service demonstration is a scalable video demonstration.

The experimentation that has been carried out during TRUST project has shown that the adaptive modulation supplies a high gain onto the link budget for radio link on multipath propagation channels with Doppler. When using adaptive modulation and adaptive error correction coding, it is expected to obtain a higher gain. The experimentation to be led in SCOUT project will allow to determine the conditions of a realistic use of the link adaptation. Moreover the interaction of the link adaptation with the scalable video transmission will be studied in order to see what sort of improvement can be provided.

3/ Transmission of scalable video sequences

Section 4 describes the interaction between the video systems developed in SCOUT A4.3.4 and A4.3.5 and the hardware validator A4.3.3. The aim is to specify a simple convenient and flexible interface which will allow scalable video systems to be demonstrated over the hardware validator, and provide meaningful results which are representative of those anticipated from a complete system.

4/ Demonstration of the middleware concept

It is clear that the reconfiguration of SDR terminals requires complex interactions between the mobile terminal and the network entities, and occasionally even the download of new software modules to be installed on the terminals. To enable mobile devices to utilize different radio access technologies and communication protocols, middleware technology will be implemented that follow the mode negotiations and switching methodology developed in TRUST. Bandwidth and delay variations emulate vertical handovers or link adaptations. Middleware requirements were already defined in SCOUT D3.1.1 and

some of these requirements as QoS monitoring in application level (measurement of download times), lightweight demands on middleware (SOAP is used here), structured storage of profile data in XML were considered in the design phase. Many other requirements in D3.1.1 were not considered as e.g. security demands in order to concentrate on essential investigations as middleware latency and QoS measurements in application level.

Authors

Partner	Name	Phone / Fax / E-mail
France Telecom R&D	Philippe du Reau	Phone: +33 1 45 29 48 41 Fax: +33 1 45 29 41 94 E-mail: philippe.dureau@rd.francetelecom.com
TTI	Zoran Golubicic	Phone: +34 942 29 12 12 Fax: +34 942 27 01 39 E-mail: golubicic@ttinorte.es
University of Bristol	David Redmill	Phone: +44 117 954 5180 Fax: +44 117 95206 E-mail: David.Redmill@bristol.ac.uk
Siemens	Eiman Mohyeldin	Phone: +49 89 636 45340 Fax: +49 89 636 45591 E-mail: eiman.mohyeldin@siemens.com
Technical University of Munich	Michael Fahrmaier	Phone: + 49 89 289-17842 Fax: + 49 89 289-17307 E-mail: fahrmaier@informatik.tu-muenchen.de
Technical University of Munich	Christian Salzmann	Phone: + 49 89 289 17838 Fax: + 49 89 289 17307 E-mail: salzmann@in.tum.de
Technical University of Munich	Peter Dornbusch	Phone: + 49 89 289-1 78 45 Fax: + 49 89 289-1 73 07 E-mail: dornbusch@informatik.tu-muenchen.de
DoCoMo Euro-Labs	Robert Hirschfeld	Phone: + 49 89 56824 221 Fax: + 49 89 56824 300 E-mail: hirschfeld@docomolab-euro.com

Table of Contents

1. Introduction	15
2. Reconfigurable RF TX chain.....	16
2.1 Demonstrator analysis	16
2.1.1 Reconfiguration possibilities	16
2.2 Specification of parameters for Scout RF Transmitter Demo	23
2.3 Description of parameters for Scout RF transmitter demo.....	24
2.4 Conclusion.....	26
3. Link adaptation on hardware validator.....	27
3.1 Introduction: objectives of the demonstration.....	27
3.2 Link adaptation	27
3.2.1 Description of the reconfiguration scenario.....	27
3.2.2 Specification of the functions to be implemented	29
3.2.2.1 Modulation.....	29
3.2.2.2 Spreading and scrambling.....	29
3.2.2.3 Slot format	30
3.2.2.4 Interleaving.....	31
3.2.2.5 Error correction code.....	34
3.2.2.6 Quality parameter estimation	35
3.2.2.7 Transmission of the configuration number	36
3.2.2.8 Reconfiguration principles.....	37
3.3 Video demonstration installation	38
3.3.1 Description of the interaction with the video data flow.....	38
3.3.2 Transmission of the video packets	38
3.3.3 Quality of transmission	39
3.4 Conditions of the experimentation	39
3.4.1 Demonstration system	39
3.4.2 Control/display environment	40
3.4.2.1 Control.....	40
3.4.2.2 Transmission control parameters.....	40
3.4.2.3 Measurements	41
3.4.2.4 Display	41
3.4.3 Phases of the demonstration	41
3.4.3.1 Transmission performances measurement	41
3.4.3.2 Link adaptation test	42
3.4.3.3 Video demonstration phases	42
3.5 Conclusion: benefits of the link adaptation.....	42
3.6 References	42
4. Video Demonstration	42
4.1 Introduction	42
4.2 Current Scalable Video Systems	42
4.2.1 Overview of H.263+.....	42
4.2.1.1 Temporal scalability	42
4.2.1.2 SNR scalability	42
4.2.1.3 Spatial scalability	42
4.2.1.4 Hybrid scalability	42
4.2.1.5 Efficiency of H.263+ layered scalability	42
4.2.2 MPEG-2 Scalability	42
4.2.3 MPEG-4 Scalability	42

4.2.3.1	Object based scalability	42
4.2.3.2	Fine granular scalability (FGS)	42
4.2.4	Future Scalable Video Systems	42
4.2.5	Challenges in future scalable systems	42
4.3	Scalable Video Application Scenarios	42
4.4	IP Protocols	42
4.4.1	Related IP protocols	42
4.4.1.1	IP-V4, IP-V6	42
4.4.1.2	Transport protocols – TCP and UDP	42
4.4.1.3	Real time protocol (RTP)	42
4.4.2	Traffic Control Techniques	42
4.4.2.1	IPv4 relative priority marking	42
4.4.2.2	IPv4 type of service (TOS)	42
4.4.2.3	Integrated Services (IS) with RSVP	42
4.4.2.4	IPv6 flow labels	42
4.4.2.5	Differentiated services (DS)	42
4.4.3	Traffic Control for Scalable Video	42
4.4.4	2 Layer scalable system	42
4.4.5	Multi-layer scalable system	42
4.4.6	Fine-grained scalable system	42
4.5	Proposed System for Scout	42
4.5.1	Scalable Video Application Scenario for SCOUT Video Demonstration	42
4.5.2	Packetized Video	42
4.5.3	Simplified SCOUT demonstration system	42
4.6	Proposed demonstration system	42
4.6.1	Connection different parts via TCP	42
4.6.2	Location of video scaling process	42
4.6.3	Information exchanged between system components	42
4.6.4	Video Source	42
4.6.5	Validator transmitter	42
4.6.6	Validator receiver	42
4.6.7	Video decoder	42
4.6.8	Graphical user interface (GUI) to video system	42
4.7	Software Interface	42
4.7.1	Video packet structure	42
4.7.2	Control data passed back from video decoder to video source	42
4.8	Objectives of the Video demonstration	42
4.8.1	Phase 1	42
4.8.2	Phase 2	42
4.9	Conclusion	42
4.10	References	42

5. Middleware Demonstrator 42

5.1	Introduction	42
5.1.1	Requirements from SCOUT D3.1.1	42
5.1.2	System Model for Reconfiguration	42
5.2	Requirements for the Middleware Demonstrator	42
5.2.1	General Objectives for the Demonstrator	42
5.2.1.1	Evaluation of Middleware Concepts	42
5.2.1.2	Measurements and Characterization of System Performance	42
5.2.1.3	Refinement of under-specified concepts	42
5.2.2	Visualization Objectives	42
5.2.2.1	Demonstration of the overall middleware concept	42
5.2.2.2	Demonstration of important key concepts of the middleware	42
5.2.2.3	Visualizing concepts	42
5.2.2.4	Demonstration of the overall middleware concept benefits	42
5.2.3	Proof of Concepts & Prototype Aspects	42
5.2.3.1	Getting data about difficulty of a concept implementation	42
5.2.3.2	Verification of Concepts	42
5.2.3.3	Validation of Concepts	42
5.2.4	Summary of the Objectives	42

5.2.5	Selection of Appropriate Demonstration Scenarios	42
5.2.5.1	Surfing the Web	42
5.2.5.2	Re-configurable Streaming Media Emulation	42
5.2.6	Analysis of Demonstrator Requirements	42
5.2.6.1	Requirements for Demonstration and Evaluation	42
5.2.6.2	Functional Middleware Requirements	42
5.2.6.3	User Interface Requirements	42
5.2.6.4	Application Scenario Functional Requirements	42
5.2.7	Demonstrator Use Cases	42
5.2.8	Technical efficiency	42
5.2.9	Demonstration of Middleware implementation	42
5.2.10	Uses Cases Description	42
5.2.10.1	Web Browsing Use Cases	42
5.2.10.2	Evaluation Use Cases	42
5.2.10.3	Combined Use Cases	42
5.2.11	User interface	42
5.3	Design Specification Overview	42
5.3.1	XML based data storage and retrieval	42
5.3.2	Main Demonstrator Function Groups	42
5.3.2.1	General Overview	42
5.3.3	Main Deployment Locations Description	42
5.4	Base Technology Overview	42
5.4.1	Differences between Services and Components	42
5.4.2	.Net	42
5.4.3	Web services	42
5.4.4	C#	42
5.4.5	XML	42
5.4.5.1	XSLT	42
5.4.5.2	XPath	42
5.4.5.3	XQuery	42
5.5	Specifications of the Main System Components	42
5.5.1	Web Browsing Application	42
5.5.2	Terminal Management	42
5.5.3	Simulation	42
5.5.4	Evaluation	42
5.5.5	Trading	42
5.5.5.1	Policies and strategies for trading	42
5.5.5.2	Trading Manager	42
5.5.5.3	Trading Protocols	42
5.5.6	Application Adaptation (Optional)	42
5.5.6.1	Sample Scenario	42
5.5.6.2	Measuring Adaptation Time	42
5.5.7	Reconfiguration	42
5.5.7.1	Reconfiguration Controller	42
5.5.8	Software Download	42
5.5.9	Profiles	42
5.5.9.1	Application and User profile	42
5.5.9.2	Measuring Network QoS	42
5.5.9.3	Measuring Application QoS	42
5.5.10	Logical Service System	42
5.6	Evaluation Specification	42
5.6.1	General Reconfiguration Timing	42
5.6.2	Profile specific Timings	42
5.6.2.1	Measuring Profile Element Change Detection	42
5.6.2.2	Measuring Profile Element Change Propagation	42
5.6.2.3	Measuring Profile Element Access	42
5.6.3	Trading specific Timings	42
5.6.3.1	Measuring Reconfiguration Computation	42
5.6.3.2	Measuring Trading Computation	42
5.6.3.3	Measuring Trading Deployment	42
5.6.4	Scalability Evaluation	42
5.6.4.1	Measuring Profile Storage Resources	42

5.6.4.2	Measuring User dependency	42
5.7	Middleware Platform.....	42
5.7.1	Hardware Deployment	42
5.7.2	Microsoft .Net, SOAP and Web services	42
5.8	Conclusion.....	42
5.9	References	42
6.	Conclusions	42

List of Figures

Figure 2-1: Simple reconfigurable UP converter. Possible problems could be function of DDS (phase noise, jitter etc) and filter realization.....	16
Figure 2-2: UP convrted DDS with relaxed filter requirements.....	17
Figure 2-3: Demonstrator structure with simple synthesizer. Two IF are necessary.....	18
Figure 2-4: Demonstrator synthesizer enable continuous switching process.....	18
Figure 2-5: Fully relaxed requirement for filter realization.....	19
Figure 2-6: Shadowing schematic for demonstrator Tx.....	21
Figure 2-7: Possible schematic for demonstrator linearization schematic.....	23
Figure 2-8: Demonstrator architecture.....	25
Figure 2-9: Power amplifier architecture.....	26
Figure 3-1: Demonstration system.....	40
Figure 4-1: Temporal scalability. The un-shaded portions represent the base layer while the shaded parts represent the enhancement layer.....	42
Figure 4-2: SNR and/or spatial scalability.....	42
Figure 4-3: Spatial scalability.....	42
Figure 4-4: A possible configuration of pictures for hybrid scalability.....	42
Figure 4-5: Object based temporal scalability.....	42
Figure 4-6: Example video applications categorized according to real-time and multi-cast constraints. The shaded region represents the main areas where scalable systems are applicable. The SCOUT video demonstration will focus on the non-real time uni-cast case.....	42
Figure 4-7: Block diagram of a typical real time bi-directional system.....	42
Figure 4-8: Block diagram of a typical non-real time (download) system.....	42
Figure 4-9: Multicast Distribution over Different Networking Systems.....	42
Figure 4-10: Typical system comprising a combination of wired and wireless networks, with scaling at the network layer.....	42
Figure 4-11: Typical system comprising a combination of wired and wireless networks, with scaling at the application layer. ¹	42
Figure 4-12: System to be used in Scout demonstrator.....	42
Figure 4-13: Block diagram of SCOUT video demonstration system.....	42
Figure 4-14: Interworking between hardware validator and video system.....	42
Figure 5-1: High-level Reconfiguration System Model.....	42
Figure 5-2: Evaluation Activity diagram.....	42
Figure 5-3: Demonstration Activity diagram.....	42
Figure 5-4: Choosing Application Service.....	42
Figure 5-5: Browsing.....	42
Figure 5-6: Switching between Devices.....	42
Figure 5-7: Evaluation Test Patterns.....	42
Figure 5-8: Evaluation Use Case.....	42
Figure 5-9: Evaluation abort use case.....	42
Figure 5-10: Displaying of Evaluation test results.....	42
Figure 5-11: Display of the system status.....	42
Figure 5-12: Manual profile manipulation.....	42
Figure 5-13: Browsing.....	42
Figure 5-14: Live Demo and Evaluation Control Center.....	42
Figure 5-15: Configuration of Test Scenario.....	42
Figure 5-16: Evaluation Result Postprocessing Control.....	42
Figure 5-17: Demonstrator components.....	42
Figure 5-18: Demonstration scenario.....	42
Figure 5-19: Trading Manger.....	42
Figure 5-20: Trading Protocol, Case one.....	42
Figure 5-21: Trading Protocol, reverse process.....	42
Figure 5-22: Trading Protocol, Case two.....	42
Figure 5-23: Failed Adaptation.....	42
Figure 5-24: Successful Adaptation.....	42

Figure 5-25: Reconfiguration Controller	42
Figure 5-26: System Adaptation	42
Figure 5-27: Example for an XML profile element	42
Figure 5-28: Logic Demonstrator	42
Figure 5-29: Hardware Structure of Demonstrator	42

List of Tables

Table 3-1: Summary characteristics of the reconfiguration scenario.....	28
Table 3-2: Modulation symbol (with $j^2 = -1$)	29
Table 3-3: UTRA FDD downlink slot formats characteristics	31
Table 3-4: Permutation patterns for the first interleaving	32
Table 3-5: Permutation pattern for the second interleaving	33
Table 3-6: Information bit rates with error correction and slot format 13..... (slot by slot reconfiguration).....	35
Table 3-8: Probability of erroneous bits in a packet.....	38
Table 3-9: Probability of erroneous decoding of the header	39
Table 3-10: ITU-R Vehicular Test Environment - Channel A	42
Table 5-1: Objectives Summary.....	42
Table 5-2: User Profile	42
Table 5-3: Application Profile	42
Table 5-4: User Profile	42
Table 5-5: Application Profile	42

Abbreviations and Definitions

A/D	Analog to Digital
BER	Bit Error Rate
CC/PP	Composite Capability/preference profile
CIF	Common Interchange Format – video resolution (352x288)
CRC	Cyclic Redundancy Check
D/A	Digital to Analog
DCT	Discrete Cosine Transform
DDS	Direct Digital Synthesizer
DMA	Direct Memory Access
DS	Differentiated Services
DSCP	Differentiated Services Control Point
EI	Enhancement Intra frame. Enhancement frame coded using intra-frame techniques.
EP	Enhancement Predictive fame. Enhancement frame coded using inter-frame prediction techniques.
FGS	Fine Grained Scalability
FIFO	First In First Out.
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IF	Intermediate Frequency
IM	InterModulation
I/O	Input / Output
IP	Internet Protocol
IS	Integrated Services
LO	Local Oscillator
LPF	Low Pass Filter
MExE	Mobile Execution Environment
MIDP	Mobile Information Device Profile
MPEG	Moving Pictures Expert Group
NL	Network Layer
OSC	Oscillator
PA	Power Amplifier
PC	Personal Computer
PD	Phase Discriminator
PDA	Personal digital assistant
PHB	Per Hop Behavior

PLL	Phase Locked Loop
PSNR	Peak Signal to Noise Ratio
QCIF	Quarter CIF – video resolution (176x144)
QoS	Quality of Service
RF	Radio Frequency
RSVP	Resource Reservation Protocol
RTCP	Real Time Control Protocol
RTP	Real Time Protocol
SDR	Software Defined Radio
SNR	Signal to Noise Ratio
SQL	Structured Query Language
TCP	Transport Control Protocol
TOS	Type Of Service Field (in IPv4).
TX	Transmitting
UDDI	Universal Discovery, Description and Integration
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
VCDRO	Voltage Controlled Dielectric Resonator Oscillator
VCSAW	Voltage Controlled Surface Acoustic Wave
VCXO	Voltage Controlled Crystal Oscillator
WLAN	Wireless Local Area Network
WSDL	Web Service Definition Language
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

1. Introduction

The present document describes the specifications of the demonstrations to be performed in the project:

- in the technology level: reconfigurable RF transmitter chain (section 2);
- in the physical layer: real time reconfigurability with a hardware validator (section 3);
- in the application level: a scalable video demonstration on the hardware validator (section 4);
- in the middleware domain: demonstration of a middleware concept for reconfiguration (section 5).

1/ Technology for reconfigurable RF Transmit chain

Like the starting point in the development of the SCOUT demo model it has been taken TRUST model for the demonstrator.

New hardware and some modifications of TRUST demonstrator were performed for achieving the compatibility with SDR requirements. Also, the phase noise and the linearity of the TX chain are improved. The appropriate architecture for the demo model, based on general-purpose components, has been designed. This shall be a suitable platform for testing the SCOUT concept.

2/ Real time reconfigurability with hardware validator

In TRUST project the hardware validator has been used to unveil some benefits of the adaptive modulation. In SCOUT project it will be used for two demonstrations:

- a real time reconfiguration of the radio interface,
- a service demonstration which will be installed onto the previous reconfiguration scenario.

The real time reconfiguration is a link adaptation to the instantaneous fluctuations of the propagation channel. The service demonstration is a scalable video demonstration.

3/ Transmission of scalable video sequences

Section 4 describes the interaction between the video systems developed in SCOUT A4.3.4 and A4.3.5 and the hardware validator A4.3.3. The aim is to specify a simple convenient and flexible interface which will allow scalable video systems to be demonstrated over the hardware validator, and provide meaningful results which are representative of those anticipated from a complete system.

4/ Demonstration of the middleware concept

Section 5 describes the requirements and design specifications of SCOUT Middleware Demonstrator. In this demonstrator we aim to demonstrate and evaluate the middleware concept. Since Middleware is usually invisible and therefore cannot be easily demonstrated as for example an application which can be presented by just using it or let interested persons play around with it, therefore our task lies on “demonstrating” and “presenting” the overall functionality of the middleware concept, the functionality of certain key concepts, visualizing of certain interesting new/key concepts and demonstrating the benefits of using Middleware technology in Mobile communication environment.

2. Reconfigurable RF TX chain

2.1 Demonstrator analysis

The objective of the demonstration is the investigation and presentation of new technology satisfying SDR Tx requirements. Main improvements in technology are dedicated to:

1. Reconfiguration possibilities
2. Shadowing channel transition
3. Phase noise improvement
4. Linearity improvement

2.1.1 Reconfiguration possibilities

RF-Tx chain is part of Software Defined Radio (SDR) System. It means that RF-Tx chain has to satisfy compatibility in sense of programmability and standards requirement with the rest of the system. RF-Tx chain consists of two parts: digital and analog. Intention in SDR system is moving all reconfigurability function from analog to digital part like inherently programmable hardware. Limitations of this process lie in digital hardware operating frequency and complexity. Good performances of digital hardware are low power consumption and high density of packing. At the existing level of technology commercially available devices enable operation of digital hardware up to RF frequencies of 400 MHz. This is data based on 1 GHz clock DDS and numerical multipliers. A/D and D/A converters have operating frequencies up to 200 MHz. IF frequency generated up to 400 MHz and with bandwidth of 300 MHz can be enough to cover all existing standards. Widest band covered (by Hyperlan 2) band between 5.475 to 5.725 GHz (250 MHz). From this reason our main goal is to put main reconfiguration function in IF (digitized) part. It means that the part of reconfiguration process, which belongs to analog part, is minimized. Best way to perform this function is up conversion IF band of 400 MHz to RF bands (at different standards). Schematic of possible configuration is given at Figure 2-1.

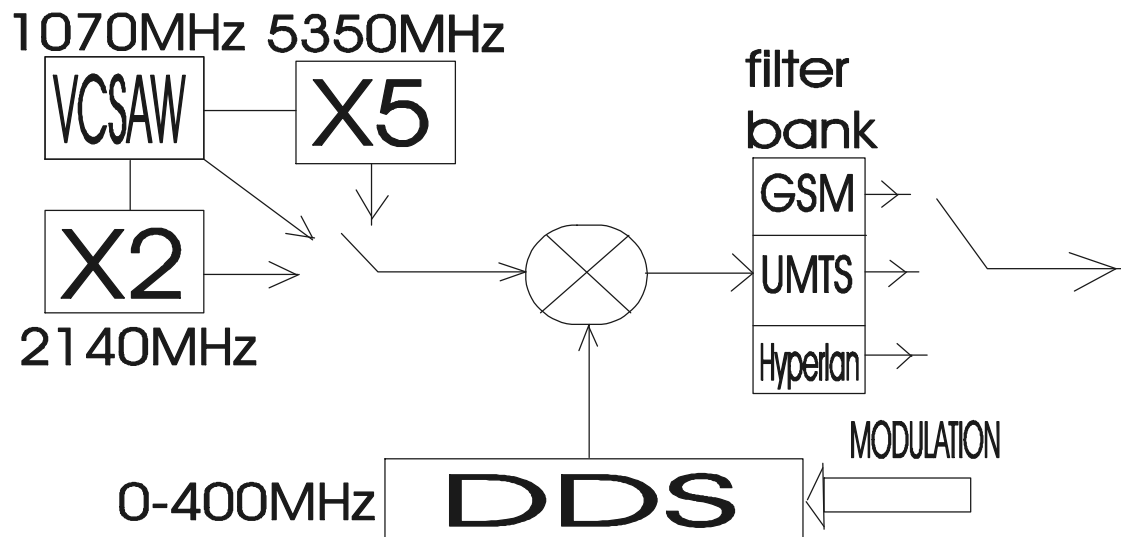


Figure 2-1: Simple reconfigurable UP converter. Possible problems could be function of DDS (phase noise, jitter etc) and filter realization.

IF signal is generated at frequency between 125 and 375 MHz. Local oscillators are realized like SAW oscillator at frequency 1.07 GHz. Second harmonic at frequency 2.14GHz is used like local oscillator for UMTS standards and fifth harmonic at 5.35GHz is used like local oscillator for Hyperlan 2. It means that IF frequencies between 125 and 375 MHz can fulfill requirements for all standards. UMTS and GSM standards are narrow band standards – 20 and 60 MHz wide. In this case filtering of higher conversion products are not critical. But in Hyperlan 2 bandwidth is 250 MHz and filtering of higher products are critical because they fall in operating bandwidth. For example if 5.5 GHz is generated by conversion 150

MHz IF signal with 5350MHz local oscillator, not only signal at 5.5 GHz but also signal of 5.65GHz ($5350 + 2 \times 150$) will be generated. This effect can be avoided with minimization of IF signal level but in that case ratio between RF and LO signal is decreased and problems with filtering LO signal are increased. Second method is incorporating adjustable notch filter at frequency of second harmonic. Essentially this problem is resident for all standards with bandwidth wider than half bandwidth of digitized IF. One solution is using LO with switching possibilities. For example if SAW oscillator has possibility to switch to 1.04GHz and 1.07GHz ($X5 = 5.2$ and 5.35 GHz) and IF is changed between 275 and 400 MHz. Second harmonic of 275 MHz is 550 MHz what means that this products is at 5750MHz what is out of band. This component can be filtered with filter covering 5.45 to 5.6 GHz bands. Second filter is intended for 5.6 to 5.75 GHz bands. It means that main problem in direct up conversion of IF signal to RF frequency (using simple microwave LO synthesizers) is realization of microwave filters.

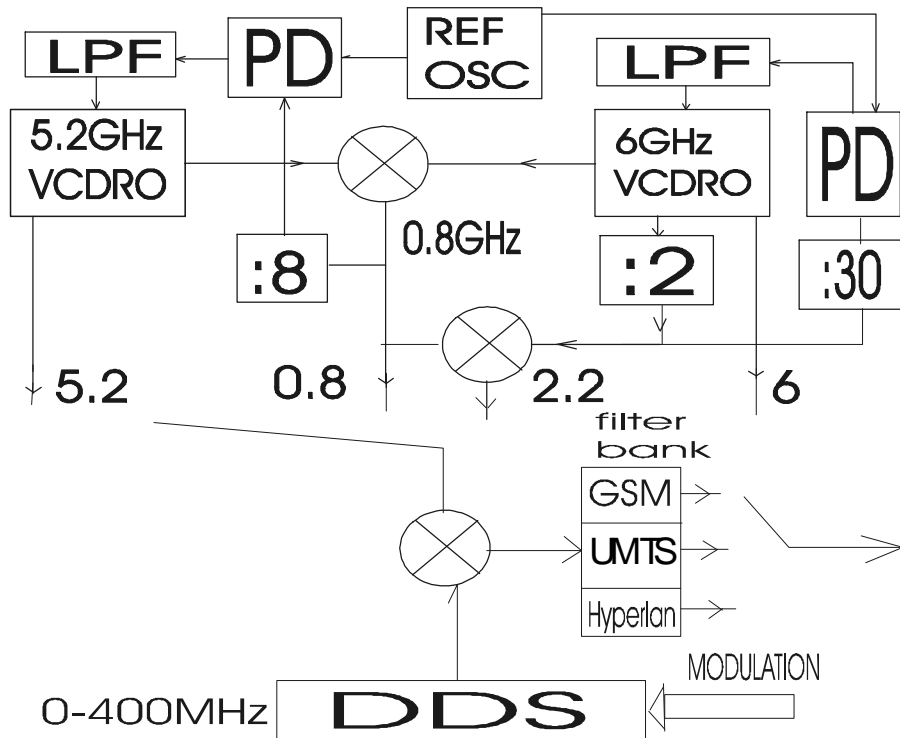


Figure 2-2: UP converted DDS with relaxed filter requirements

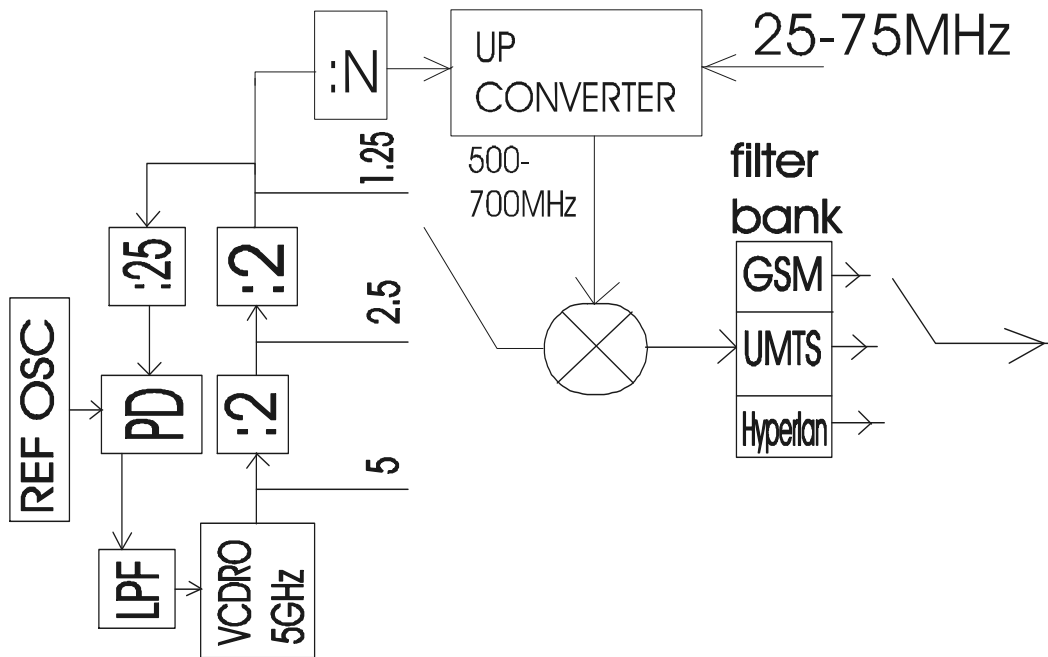


Figure 2-3: Demonstrator structure with simple synthesizer. Two IF are necessary

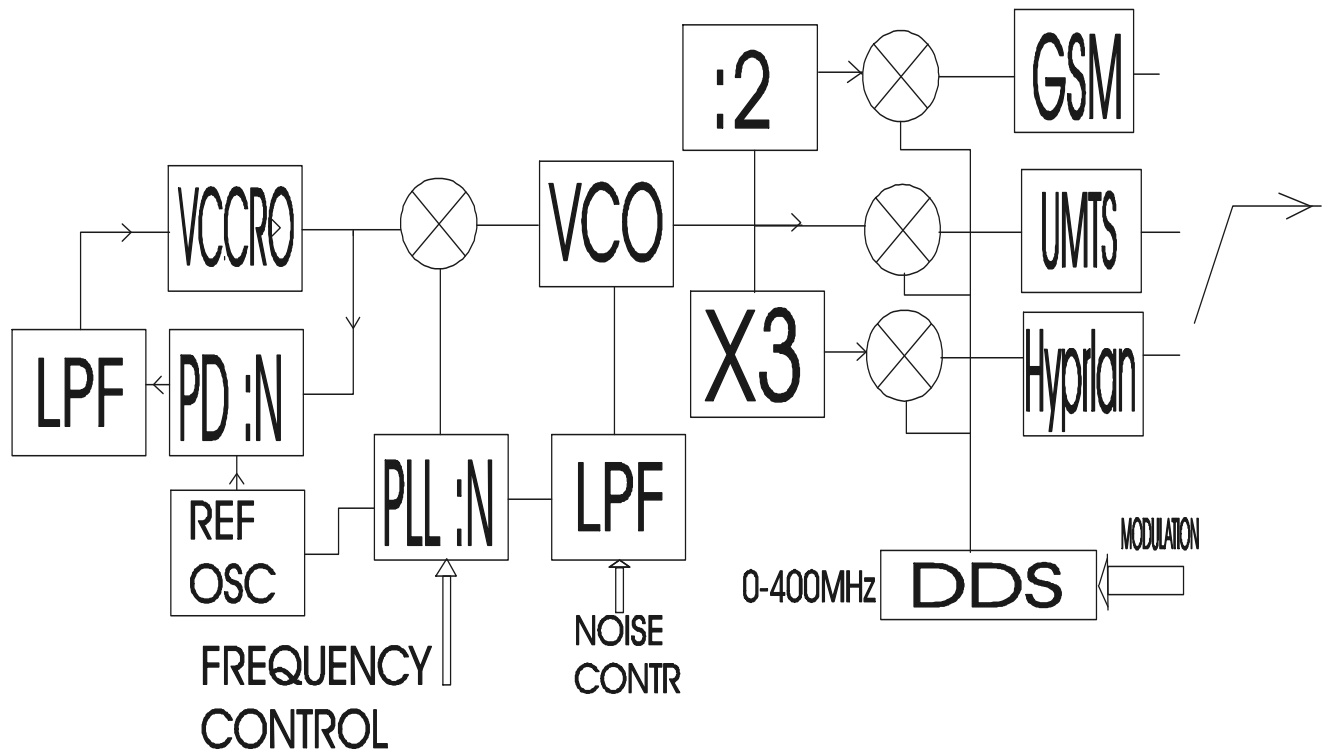


Figure 2-4: Demonstrator synthesizer enable continuous switching process.

RF application but it is more flexible because it is connected with host processor power over DMA devices.

Solution for demonstrator model microwave LO is presented at Figure 2-3. Main oscillator is VCDRO operating at 5GHz. This signal is mixed with IF signal 500 to 700MHz. Signal in band 1800 to 2000 MHz is obtained, by mixing same IF with signal at 2.5GHz (5:2). For GSM signal near 350 MHz is necessary (to be mixed with 1250MHz signal..

In the case that DDS chip is not available (data sheets are now only preliminary) signal from frequencies 500 to 700 MHz can be generated by up converting signal at frequency 25 to 75 MHz (digital up converters up to 80 MHz are available) with one of the subcarrier at 625MHz (1250:2) or synthesized 575 MHz signal. Like the subcarrier for 350 MHz can be used half frequency of one subcarrier (625MHz:2). Signals in band 25 – 75 MHz is produced by upconverting 10MHz signals generating from host computer with multi I/O board. This solution enables full flexibility in development process.

Advance of this schematic is minimum transition time and realization of frequency transition with fixed oscillator what mean that same oscillator can be used in main and shadow channel. There are no programming of PLL. Frequencies are chosen to decreased IM products in mixers and relaxed requirement for filters.

2.1.2 Shadow channel

Channel shadowing presents the best way to perform channel switching without losses in transmitted bits. Also spurious generation can be minimized. In that case we can distinguish two cases:

- channel changing inside the same standard
- channel changing between two different standard

Channel switching between two different standards is a well known case when the terminal contains hardware dedicated to each standard particularly. Usually this hardware is separated with diplexer circuits and shadowing is achieved with simple switching on two standards simultaneously during transition period.

Transition between channels inside the same frequency standard can be shadowed using distributed reconfigurable system. Example of such system is presented at Figure 2-6. Theoretically shadowing can be realized for full mobile bandwidth (if other components allow operation in full bandwidth) but because bandwidth is separated in two bands we have to realize two shadow channels for lower and higher band separately. Local oscillator system is the same for upper and lower bandwidth and switching process are not needed (except switching on and switching off power amplifier). Local oscillators are realized like fixed oscillator and phase noise performances can be optimized. Carrier distribution can be performed at digital circuit only. Standard switching is performed with switching power amplifier. This is normal procedure in multistandard operation for optimization power consumption. Time necessary for channel changing is negligible comparing to symbol interval. Main part of time loosed in transition process is the time necessary for amplifiers switching. For this reason shadowing channel enable information flow over two paths during amplifiers set up. Controlled process of power amplifiers switching suppresses spurious generation during transition period.

In previous schematic, used power amplifiers are realized like balanced amplifiers. Power of each amplifier used in balanced configuration is two times less than power of amplifier used in single amplifier architecture. It means that hardware of power amplifier is not doubled. Balanced configuration is also useful for some linearization method, insensitivity to mismatching etc. Taking into account that oscillator circuit is not doubled we can conclude that existing schematic will not double the hardware of active components. These components are not only main power consumer but also they are the components with highest prices. Doubled components are passive component mixers and appropriate filters what is relatively small increasing in overall prices.

Scenario for channel switching between bands is very simple. During normal operation both amplifier in one standard are feed with same input signal. If we decide to activate other standard at the output of D/A converter is generated signal with appropriate frequency. Position of switcher is chose to activate another standard with shadow chain.

Scenario for channel switching inside the channel is more complicated. During usual operation both amplifiers are in function with appropriate phase. If we decide to change channel one amplifier is continuously switched off. Frequency on shadowing D/A converter is changed and switched off amplifier is feed with new channel. In this moment both channels transmits. After that amplifier feed with old channel is switched off and his input is connected to new channel. Amplifier is now again switch on and full power at new channel is set up. This algorithm is useful not only for changing channel different in frequency but also for changing channel at same carrier with different codes. Channel reconfiguration in this way has advantage over single amplifier architecture because generation of two signals through same amplifier need significant decreasing in power if linearity of transmission is necessary. Two separate amplifiers require less power reduction in transmitted power during transition period.

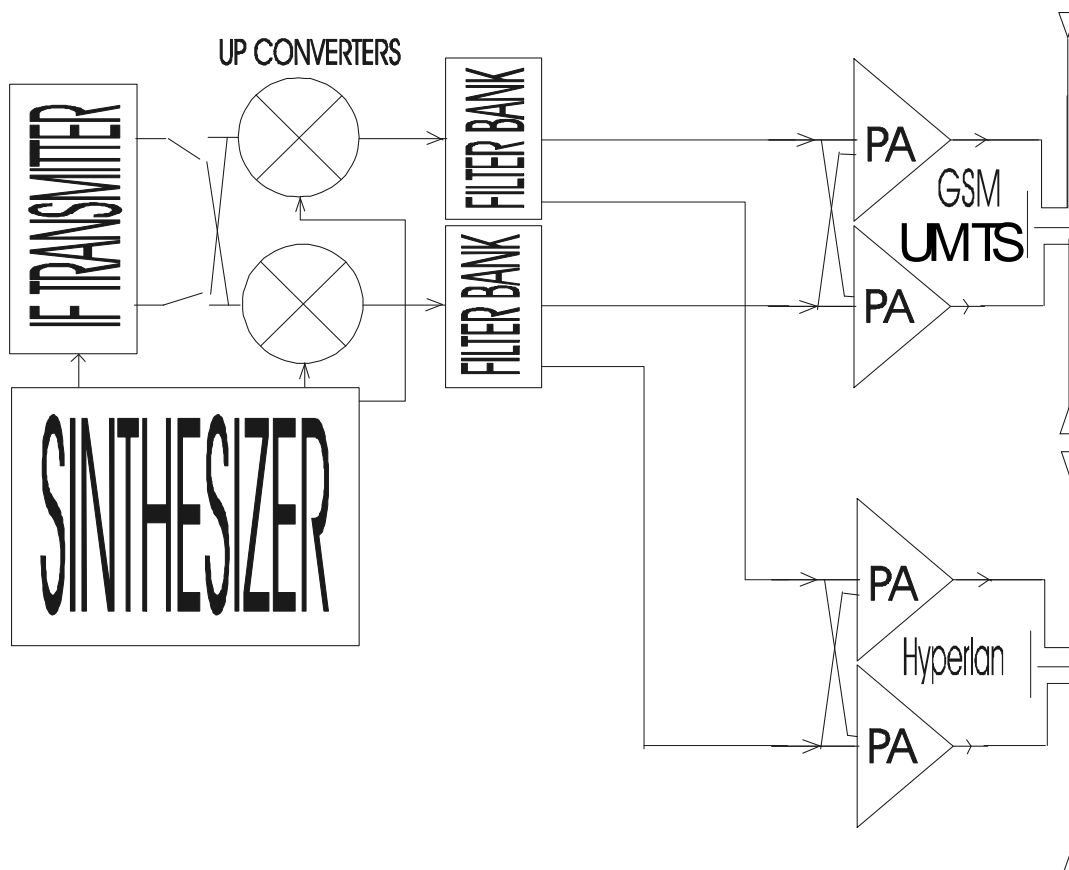


Figure 2-6: Shadowing schematic for demonstrator Tx

2.1.3. Phase noise improvement

One of the objectives of demonstrator model is phase noise improvement. Our objective is to generate so good phase noise of RF local oscillators to move responsibility for phase noise to digital IF signals. In order to achieve good phase noise of local oscillators at microwave frequency we have to use oscillator with high Q resonant elements locked by PLL to reference oscillator. Reference oscillator, what is usually quartz crystal oscillator has very good phase noise at lower frequency offsets – close to carrier. Noise floor at larger frequency is also with low level but it increases by multiplying in PLL. This multiplied level at certain offset frequency overcome noise level of free running oscillator. If oscillator is realized with high Q resonant element, offset frequency where these two values are the same is closer to carrier. In that case optimum phase noise can be obtained with smaller loop bandwidth.. Unfortunately these type of oscillators are fixed frequency oscillators because high Q resonators are elements with fixed dimensions. Some advantages can be expected in the future using MEMS technology but now we shall take into account resonators with fixed dimensions.

One of the main limitations in channel reconfiguration switching time is very narrow bandwidth of PLL necessary for channel resolution. Problem of channel resolution is moved to digital part and very good frequency resolution can be obtained inside 10 MHz frequency step. Eventually channel spacing at higher frequency can be obtain using higher PLL reference and wider PLL bandwidth. Higher bandwidth enable faster locking but phase noise inside the bandwidth of the loop origin from multiplied phase noise of reference oscillator. If the LO used for channel adjusting is high frequency oscillator multiplication number is high and degradation of phase noise inside PLL is not acceptable. For this reason it is suitable to perform fine channel adjusting at some IF at lower frequency when degradation of phase noise is not significant and later to up convert signal at highest frequency.

Apart from analog level subject of investigation will be the level of phase noise and spurious components generated in digital part. This level can vary a lot depend the method and hardware used for signal generation. Some data about phase noise limitation and spurious level is presented in data sheets for DDS chip.

2.1.4 Linearization in demonstrator model

In the demonstrator model five types of linearization methods can be applied:

- predistortion
- feedback
- feedforward
- signal processing
- operation point and matching circuit adjustment

Predistortion method is method suitable for incorporation in digital part of demonstrator. Ordinary demo signal and predistorted demo signal can be deployed in computer memory. Because DMA channel is applied signal of enough length can be recorded.

Feedback method is based on amplitude and phase correction in input signal according measured difference between input and output waveform. According measured difference in signals, Cartesian (I,Q) or vector (phase and amplitude) modulation is applied to correct input signal in order to minimize error. Transfer characteristic in feedback loop can be achieved through analog error amplifiers and filters or through digital processing part. Transfer characteristic created by digital part is programmable so flexibility in predistortion can be incorporated including adaptive filtering, nonlinear characteristic with programmable memory etc. This type of linearization is essential when amplitude and phase modulation of carrier is applied directly to output stage (power amplifier). This type of modulation is applied for optimization of power consumption and amplifier efficiency when non-constant envelope signal is applied. In this case prediction of delay and equalization can be made only by digital processing feedback.

Difference between feedforward and feedback method is based on correction point. Feedforward method applies correction signal to output port instead to input port applied by feedback. Feedforward method inject error signal between input and output waveform to output port but with opposite phase. This is the way for cancellation of error signal. Main effort is dedicated to calculation appropriate time delay and error amplifier gain for optimum cancellation.

Processing linearization is one of the main advantages of using balanced configuration power amplifier presented at Figure 2-7. Processing linearization can be achieved by CALLUM, LINC, LIST or any other signal processing linearization techniques. Vector adding of two constant envelope signals passing through independent amplifier chains can produce wanted amplitude and phase modulation. It enables amplifier operation in saturation (nonlinear) regime, resulting linear output and high efficiency.

Operation point and matching circuit adjustment is method mainly intended for efficiency increasing but could be use for linearity improvement. This method is more oriented to circuit design and overcome main target of demonstrator model.

All methods are fully integrated in DSP control. For demonstrator model separate PC card with A/D converter is used for monitoring function. Software will be incorporated in PC for flexibility and development.

Linearity of mixer can be controlled by using controlled biasing and power level of the input signal. Also matching circuit can be configurable towards minimum distortion. Some type s of feedback circuits are also possible but these circuits are relatively complex and can be applied only in special situation when other methods are not enough.

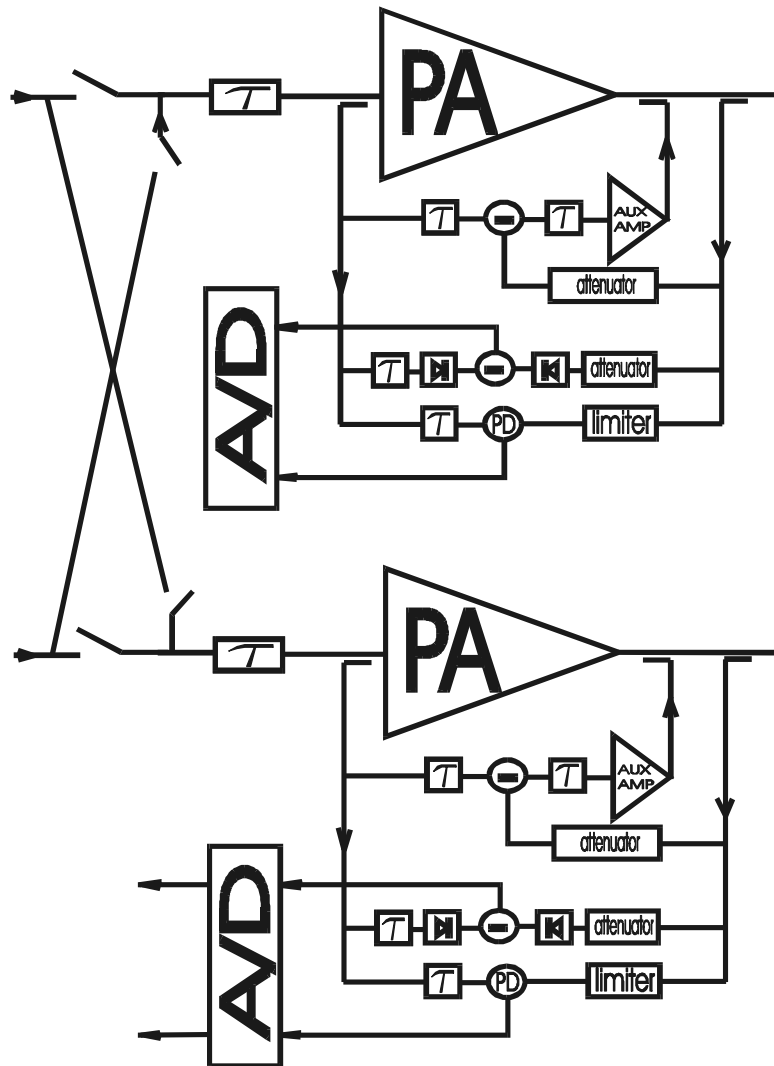


Figure 2-7: Possible schematic for demonstrator linearization schematic.

Description of concrete solution for demonstrator realization is given in next pages. Because resources for demonstration model are limited (mainly in time) only a particular linearization method will be realized.

2.2 Specification of parameters for Scout RF Transmitter Demo

In this section it is established the specifications for the Scout RF Transmitter Demonstrator.

1. Operation Standard
 - GSM
 - UMTS
 - Hyperlan2

2. Carrier Frequencies – According to Standards
3. Signal Bandwidth – According to Standards
4. Output Power >23dBm in all regimes
5. Switching Speed <200ns
6. Operational Mode Single Channel + Shadow
7. Frequency Stability – dedicated by external source
8. System Phase Noise < 10dB
9. Frequency Step < Half Channel Bandwidth (Depend on standard)
10. Demo Signal Generation – PC based D/A card
11. Linearisation Improvement Method – Predistortion of input signal
12. Status Acquisition – by PC based A/D card
13. Parameter Control – Microcontroller and Host PC

2.3 Description of parameters for Scout RF transmitter demo

The demonstration model consists of:

- Host PC including A/D and D/A card
- Two up converters for conversion signal in band 10 to 20 MHz to frequency 50 to 60 MHz and 350 to 360 MHz. It needs two local oscillators with frequencies of 70 and 300MHz (VCXO PLLs).
- DDS synthesizer for frequency 200 to 400 MHz.
- 5GHz PLL locked VCDRO. Frequency of 5GHz is divided by 2 and 2.5 GHz signal is obtained for LO signal intended for UMTS. This signal is divided by two and signal of 1.25 GHz is used like LO signal for GSM.
- Two filter bank consisting of filters intended for particular standard.
- Two balanced amplifier - one for GSM and UMTS and other for Hyperlan2. standards.
- Diplexer

Signals at output frequencies are obtained by mixing 5 or 2.5 GHz LO signals with 0.5 to 0.7 GHz IF or 1.25GHz LO signal with 350-360MHz IF.

Frequencies 500-700MHz are obtained by mixing fixed IF of 350-360MHz with LO of 850 to 1050MHz. Last signal is obtained by mixing DDS signal (200 to 400 MHz) with 1.25GHz LO.

Signal covering GSM is obtained by direct mixing 350 to 360 MHz signals, passing through second mixer transparently (without conversion), with 1.25GHz LO. GSM can be obtained by directly DDS modulation.

Schematic of power amplifier is essentially same like in TRUST with possibilities that each amplifier in balanced configuration can be driven separately. This is enabled with additional switch controlled by microcontroller.

Amplifier parameters – phase and amplitude distortion are monitored with A/D card included in host PC. According that PC calculates predistortion in input signal. Schematic is shown in Figure 2-8. A detailed schematic of the amplifier is presented in Figure 2-9.

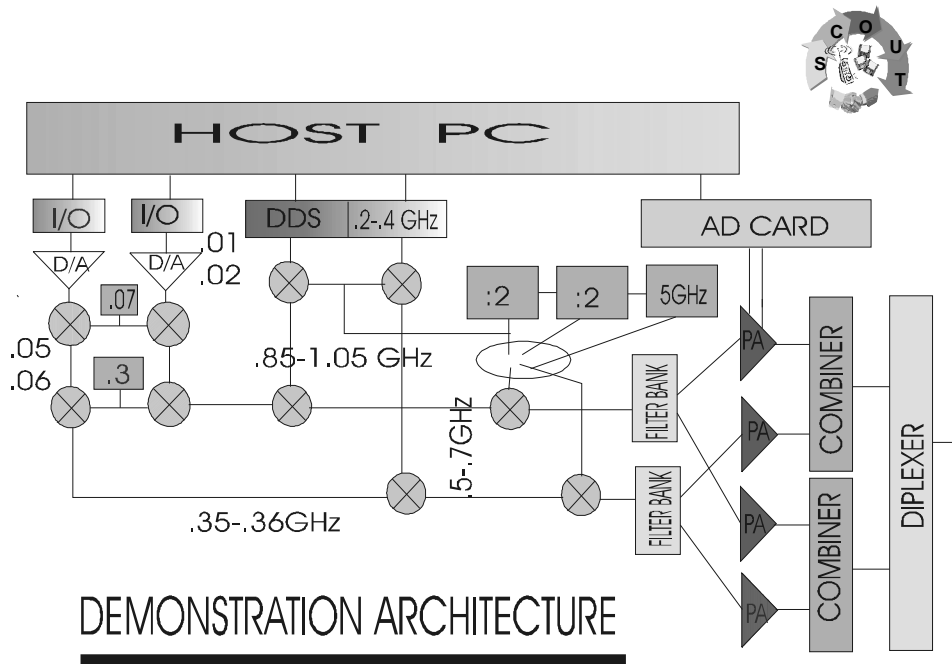


Figure 2-8: Demonstrator architecture

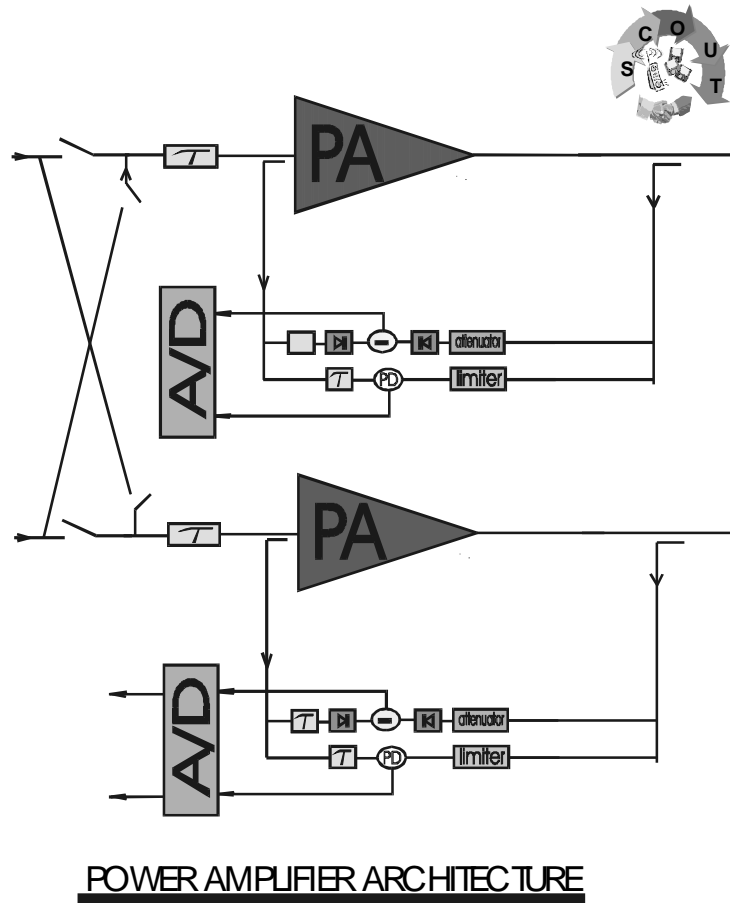


Figure 2-9: Power amplifier architecture

2.4 Conclusion

The demonstration will show shadow transition between standards and between carriers inside the same standard. By time control output power spurious generation will be eliminated. The information losses during transition period will be avoided in the case when two different standards will be changed. Fine frequency step will be demonstrated without phase noise degradation. Signal predistortion will be applied thankfully to introduced processing power.

3. Link adaptation on hardware validator

3.1 Introduction: objectives of the demonstration

A radio transmission hardware validator has been developed by France Telecom R&D in order to investigate the capabilities of the software radio techniques at the physical level. This hardware validator allows for a one-way radio transmission reconfigurable in real time without any interruption of the transmission. By this validator, it is possible:

- to test whether some proposed methods are usable in a realistic manner,
- to demonstrate in which range the transmission parameters can be reconfigured in real time,
- to show the practical impact of the reconfiguration onto the quality of transmission for the user.

In TRUST project the validator has been used to unveil some benefits of the adaptive modulation [3-5]. In SCOUT project it will be used for two demonstrations:

- a real time reconfiguration of the radio interface,
- a service demonstration which will be installed onto the previous reconfiguration scenario.

The real time reconfiguration is a link adaptation to the instantaneous fluctuations of the propagation channel. The service demonstration is a scalable video demonstration.

The aim of the current chapter is:

- to describe the link adaptation scenario,
- to describe the interaction with the video demonstration,
- to show the stages of the experimentation.

3.2 Link adaptation

3.2.1 Description of the reconfiguration scenario

The purpose of the link adaptation experimentation is to demonstrate that the average information rate can be increased or that the link budget can be improved when using modulation and channel coding that are changed in real time according to the quality of the transmission channel. Implementing a link adaptation system requires:

- to define a quality parameter that characterizes the transmission channel quality with accuracy and can be measured in a short time with respect to the reconfiguration period,
- to use modulation types and error correction codes different enough to be selected by the quality parameter in an effective manner and to bring significant performance improvement when changing from one configuration to another.

In TRUST project the real time reconfiguration selected for demonstration is an adaptive modulation on a wideband CDMA downlink. The system is UTRA FDD with appropriate adaptations. The reconfiguration is between BPSK, QPSK and 16QAM modulations. The reconfiguration is performed on a frame by frame basis. The error correction coding is fixed: it can be used or not, but it is not intended to change its use in real time. The quality parameter chosen for deciding the configuration changes is the signal to noise ratio estimated in real time from the received signals. The feedback signals from the receiver to the transmitter are transmitted by wired connections.

In SCOUT project the link adaptation will be improved in the points that follow.

1/ Reconfigurable error correction coding

The aim of the link adaptation is to guarantee some quality (e.g. a given bit error rate) on the recovered information signal when the propagation channel characteristics fluctuate. The adaptive modulation joined with a fixed error correction coding allows for reaching that result, but in fact the obtained quality is much better than the wanted quality during the most part of the time. An adaptive error correction coding will be added to the adaptive modulation instead of the fixed coding in order to follow hard the guaranteed quality while increasing the average information rate.

2/ Slot by slot reconfiguration

With a propagation channel whose characteristics vary quickly during the time, it is needed to reconfigure the transmission very often in order to follow the fluctuations of the channel. The reconfiguration will be carried out on a shorter time interval than the frame. Practically it will be the slot interval which is the shortest time interval in the frame with some control bit fields, or a multiple of the slot.

3/ Anticipation of the signal to noise ratio

The estimation of the signal to noise ratio is performed in the current configuration time interval and applied in the next configuration interval. When the channel fluctuates quickly, the performances of the link adaptation are reduced. In order to improve the adaptation, some methods will be developed for estimating the future evolution of the channel.

4/ Return way

The information returned to the transmitter by the receiver is a configuration number to be used for the next configuration period. The return is carried out by a wired connection. In the reality the return way is a radio transmission way with the same kind of perturbations as for the the main way. In order to simulate a return way, some perturbations and some delay will be applied to the information returned to the transmitter.

In order to build the reconfiguration scenario we consider the UTRA FDD physical layer and extract some characteristics from it. Modulation that is used by UTRA is QPSK, not BPSK nor 16QAM. Then some adaptations on the UTRA specifications are necessary.

First the physical channels used by a UTRA system are some dedicated physical channels transporting the information data and the control data intended for a user, and some common physical channels for access, control, synchronization or various indications and operations (see [3-1]). We choose to implement a dedicated physical channel only. In the uplink data and control bits are mapped separately into I and Q channels; that mapping cannot be adapted to a BPSK or 16QAM scheme. On the other hand in the downlink data and control bits are put together into the dedicated physical channel DPCH, and then they are split alternately between I and Q channels; it is very easy to translate the process to a BPSK scheme by not splitting the bits, but the introduction of 16QAM modulation requires to define another process. Moreover in the downlink, a common pilot channel CPICH may be used as a phase reference for the channel DPCH. We decide to implement it for making the demodulation easier.

Then UTRA FDD is a wideband CDMA system with spreading on I and Q channels, and with a complex scrambling available for QPSK modulation. In BPSK there is no difficulty for using the spreading, and the complex scrambling has to be replaced by a real scrambling. For using a 16QAM modulation we have to generalize the notions of spreading and scrambling.

The general characteristics of the transmission can be summarized as follows (Table 3-1):

Standard	UTRA FDD with adaptations
Link type	downlink with a dedicated physical channel DPCH and a pilot channel CPICH
Access type	continuous wideband CDMA
Chip rate	3.84 Mchip/s
Frame	10 ms duration; 15 slots of 2560 chips
Pulse-shaping filter	root-raised cosine with roll-off 0.22
Modulation	BPSK, QPSK, 16QAM
Spreading and scrambling	as defined in UTRA FDD with adaptations
Reconfiguration recurrence	slot by slot
Reconfiguration parameter	signal to (noise+interferers) ratio

Table 3-1: Summary characteristics of the reconfiguration scenario

At first view we could think that a simple BPSK/QPSK/16QAM scheme without spreading and with some TDMA transmission is easier to be implemented than a continuous W-CDMA. In fact the advantages of the UTRA FDD for the demonstration are as follows:

- the scrambling length allows for carrying out the frame and slot synchronization (remember that we do not implement any broadcast, access or synchronization channel);
- the frame contains pilot words usable for coherent demodulation;
- the spreading allows for performing a reconfiguration without any guard time (when reconfiguring the modulation, the impulse response duration of the filters can disturb some chips, but not the bits if the spreading factor is high enough).

3.2.2 Specification of the functions to be implemented

The functions to be implemented are:

- the multiplexing and channel coding,
- the spreading and modulation,
- the corresponding operations at the reception,
- the quality parameter estimation,
- the transmission of the configuration number.

It is not intended to implement the full mapping (multiplexing and channel coding operations) that is defined in UTRA standard, but only the functions that are necessary for the demonstration. These necessary functions are:

- the channel coding for error correction,
- the interleaving,
- the mapping of the data onto the frame of the physical channel.

Channel coding and interleaving will be only used when a very low bit error rate is needed, e.g. for video transmission.

3.2.2.1 Modulation

The modulation for UTRA FDD is defined in document [3-3]. In order to extend the modulation types to BPSK and 16QAM cases, we define a sequence of complex symbols from the sequence of bits to be transmitted. If $\mathbf{a}(n)$ is the bit number n whose possible values are $+1$ and -1 , and if $\mathbf{b}(n)$ is the complex symbol, the mapping from $\mathbf{a}(n)$ sequence to $\mathbf{b}(n)$ sequence is as follows (Table 3-2):

Modulation	Number of bits per symbol	$\mathbf{b}(n)$
BPSK	1	$\mathbf{a}(n)$
QPSK	2	$\mathbf{a}(2n) + j.\mathbf{a}(2n+1)$
16QAM	4	$\mathbf{a}(4n).[1 + 1/2.\mathbf{a}(4n+1)] + j.\mathbf{a}(4n+2).[1 + 1/2.\mathbf{a}(4n+3)]$

Table 3-2: Modulation symbol (with $j^2 = -1$)

3.2.2.2 Spreading and scrambling

The spreading and scrambling for UTRA FDD are defined in document [3-3]. The spreading factor is defined with respect to the symbols sequence, not to the bits sequence. The channelization code \mathbf{c}_{ch} and the scrambling code \mathbf{c}_s of the UTRA FDD standard for the downlink are kept without any modification for the three modulation cases. The complex symbols $\mathbf{b}(n)$ are multiplied by the real channelization code and by the complex scrambling code in order to deliver the complex $\mathbf{I}+j\mathbf{Q}$ symbols that are transmitted:

$$\mathbf{I}+j\mathbf{Q} = \mathbf{b}(n).\mathbf{c}_{ch}.\mathbf{c}_s$$

The channelization codes are Orthogonal Variable Spreading Factor (OVSF) codes defined by recurrence in the following manner. The chip $n^{\circ} i$ (for $i = 0$ to $2^n - 1$) of the code $c\{2^n, p\}$ with the number p (for $p = 0$ to $2^n - 1$) and with the length 2^n (where n is an integer) is:

- for even p

$$c\{2^n, p\}(i) = \begin{cases} c\{2^{n-1}, p/2\}(i) & \text{for } i = 0 \text{ to } 2^{n-1} - 1 \\ c\{2^{n-1}, p/2\}(i - 2^{n-1}) & \text{for } i = 2^{n-1} \text{ to } 2^n - 1 \end{cases}$$

- for odd p

$$c\{2^n, p\}(i) = \begin{cases} c\{2^{n-1}, (p-1)/2\}(i) & \text{for } i = 0 \text{ to } 2^{n-1} - 1 \\ -c\{2^{n-1}, (p-1)/2\}(i - 2^{n-1}) & \text{for } i = 2^{n-1} \text{ to } 2^n - 1 \end{cases}$$

with the starting point $c\{1, 0\}(0) = 1$

The channelization code c_{ch} is the code $c\{S, n_{ch}\}$, where S is the spreading factor and n_{ch} is the code number that is assigned to the mobile station by the base station.

The scrambling code c_s is built up from two real codes c_{s1} and c_{s2} . The chip $n^{\circ} i$ (for $i = 0$ to 38399) of the code $c_s n^{\circ} n$ (for $n = 0$ to $2^{18} - 2$) is:

$$c_{s,n}(i) = c_{s1,n}(i) + j \cdot c_{s2,n}(i)$$

The codes $c_{s1,n}$ and $c_{s2,n}$ are **38400**-chip segments of a Gold sequence. As their lengths are equal to the frame length, they allow to obtain the frame and slot synchronization inside the receiver.

The codes $c_{s1,n}$ and $c_{s2,n}$ are built up from the auxiliary sequences x , y , z_n in the following manner. The chips $n^{\circ} i$, for $i = 0$ to 38399 , are expressed by the relationships:

$$\begin{aligned} c_{s1,n}(i) &= 1 - 2z_n(i) \\ c_{s2,n}(i) &= 1 - 2z_n([i+131072] \text{ modulo } 2^{18} - 1) \end{aligned}$$

with

$$z_n(i) = x([i+n] \text{ modulo } 2^{18} - 1) + y(i) \text{ modulo } 2$$

and

- for $i = 0$

$$x(0) = 1$$

$$y(0) = 1$$

- for $i = 1$ to 17

$$x(i) = 0$$

$$y(i) = 1$$

- for $i = 18$ to $2^{18} - 1$

$$x(i) = x(i-11) + x(i-18) \text{ modulo } 2$$

$$y(i) = y(i-8) + y(i-11) + y(i-13) + y(i-18) \text{ modulo } 2$$

3.2.2.3 Slot format

The UTRA FDD frame consists of 15 slots. Each slot in a downlink DPCH channel contains one or two data fields, a pilot word, some bits for the power control and a TFCI (Transport Format Combination

Indicator) field (see [3-1]). The slot structure will be kept in BPSK and 16QAM cases. The pilot word will be used for the coherent demodulation. The power control bits are not useful here because the transmission is unidirectional. The TFCI bits are optional bits intended to be used in the case of several simultaneous services (it is not the case here). As the number of bits per symbol is different in BPSK, QPSK and 16QAM, the standard bit pattern for the pilot word cannot be kept without modification in BPSK and 16QAM. In 16QAM we shall add some extra bits to fill in the pilot field. In BPSK half the pilot bits would have to be suppressed, but the power control field and some bits of the TFCI field can be used for transmitting them.

For the demonstration, only normal slots, no compressed slots will be used. For example the slot standard formats 12 and 13, which allow for data rates of 45 ksymbol/s and 105 ksymbol/s respectively, seem well adapted for a demonstration (Table 3-3). In particular slot format 13 supplies a sufficient rate for a video transmission when using an error correction code on the transmission channel.

Slot format number	12	13
Spreading factor	64	32
Number of symbols per slot	40	80
Number of data symbols	30	70
Number of pilot symbols	4	4
Number of power control symbols	2	2
Number of TFCI symbols	4	4
Channel symbol rate (ksymbol/s)	60	120
Data symbol rate (ksymbol/s)	45	105
Data rate in BPSK (kbit/s)	45	105
Data rate in QPSK (kbit/s)	90	210
Data rate in 16QAM (kbit/s)	180	420

Table 3-3: UTRA FDD downlink slot formats characteristics

3.2.2.4 Interleaving

For transmitting pictures it is necessary to use an error correction code in order to ensure a low bit error rate. An interleaving must be used between error correction coding and modulation in order to split the error bursts in the receiver. The interleaving that is defined in UTRA FDD standard consists of a first interleaving that interleaves the bits inside blocks of data whose duration is a multiple of 10 ms and a second interleaving that interleaves the bits inside 10-ms blocks (see [3-2]). To compare the performances obtained on a given transmission channel for the three modulation cases, it is desirable that the interleaving duration can be kept the same in the three cases. A symbol consists of 1 bit in BPSK, 2 bits in QPSK, 4 bits in 16QAM. Two inferences have to be drawn from using an interleaving:

- when changing the modulation, the number of bits involved in the interleaving is changed;
- no reconfiguration is possible inside an interleaving period.

Then for a reconfiguration period shorter than the frame duration, the first interleaving must not be used and the second interleaving has to be adapted to blocks whose duration is equal to the reconfiguration period.

When using an interleaving depth that is greater than the frame duration, the reconfiguration must not be performed on a frame by frame basis, but on an interleaving depth basis. In that case, as the demonstration uses only one DPCH channel but no synchronization channel, it is necessary to number the frames when using an interleaving whose duration is greater than the frame duration. A part of the TFCI field in the frame will be used to number the frames, and consequently to allow for deinterleaving.

When using an interleaving depth short enough with respect to the fluctuations of the propagation channel, interleaving is not strictly necessary, because the channel is approximately stationary during a configuration period.

Interleaving and link adaptation by adaptive modulation and reconfigurable error correction are rival processes. If the interleaving duration is increased, the fades and the transmission defects are more and

more screened, and the link adaption cannot be very effective. For the most part of the trials the interleaving depth will be chosen equal to the slot duration in preference to longer durations, or no interleaving will be applied.

1/ Interleaving depth greater or equal to the frame

For an interleaving depth greater or equal to the frame, the rules of interleaving that are defined in UTRA FDD standard will be maintained but applied to a number of consecutive bits that depends on the modulation. Let P be the number of data bits in a frame, and let N be the number of consecutive frames to be interleaved together. Then the first interleaving spreads over a block of $N \cdot P$ bits and the second interleaving spreads over a block of P bits.

The first interleaving is defined in the following manner. Let $x(n)$ and $y(n)$, for $n = 0$ to $N \cdot P - 1$, be the sequences of bits respectively at the input and at the output of the first interleaver. Let k be a permutation of the integers from 0 to $N - 1$, that assigns a particular integer $k(j)$ in $[0; N - 1]$ as image of each integer j in $[0; N - 1]$. Then any integer n can be expressed by a single manner in the form:

$$n = jP + i$$

where i and j are integers, and i satisfies the requirement:

$$0 \leq i < P$$

The output bit $y(n)$, for $n = 0$ to $N \cdot P - 1$, is connected to the input sequence by the relation:

$$y(n) = x(i \cdot N + k(j))$$

The permutation rule of the first interleaving depends on N . The permutation pattern of the natural sequence $\{0, 1, \dots, N - 1\}$ is (Table 3-4):

Interleaving depth	N	Permutation pattern
10 ms	1	{0}
20 ms	2	{0,1}
40 ms	4	{0,2,1,3}
80 ms	8	{0,4,2,6,1,5,3,7}

Table 3-4: Permutation patterns for the first interleaving

The second interleaving is defined in the following manner. Let $x(n)$ and $y(n)$, for $n = 0$ to $P - 1$, be the sequences of bits respectively at the input and at the output of the second interleaver. Let k be a permutation of the integers from 0 to 29 , that assigns a particular integer $k(j)$ in $[0; 29]$ as image of each integer j in $[0; 29]$. The integer P is expressed by a single manner in the form:

$$P = 30Q - R$$

where Q and R are integers, and R satisfies the requirement:

$$0 \leq R \leq 29$$

The set of the integers from 0 to $P - 1$ is partitioned into 30 consecutive intervals. The interval number j , for $j = 0$ to 29 , is:

$$[A(j); A(j+1) - 1]$$

where $A(j)$ is defined by recurrence:

$$A(j) = \begin{cases} A(j-1) + Q & \text{if } k(j) \leq 29 - R \\ A(j-1) + Q - 1 & \text{if } k(j) > 29 - R \end{cases}$$

with the starting and stopping points:

$$\begin{aligned} \mathbf{A(0)} &= \mathbf{0} \\ \mathbf{A(30)} &= \mathbf{P} \end{aligned}$$

Then any integer \mathbf{n} , for $\mathbf{n = 0}$ to $\mathbf{P-1}$, belongs to some interval $\mathbf{n^\circ j}$ and to nothing else, and can be expressed by a single manner in the form:

$$\mathbf{n = A(j) + i}$$

where \mathbf{i} is an integer that satisfies the requirement:

$$\mathbf{0 \leq i < A(j+1) - A(j)}$$

The output bit $\mathbf{y(n)}$, for $\mathbf{n = 0}$ to $\mathbf{P-1}$, is connected to the input sequence by the relation:

$$\mathbf{y(n) = x(30i + k(j))}$$

The permutation rule $\mathbf{k(j)}$ of the second interleaving is defined by the permutation pattern of the natural sequence $\{\mathbf{0,1,...29}\}$. That pattern is (Table 3-5):

$\{\mathbf{0,12,25,6,18,3,15,26,9,22,2,13,24,7,19,4,16,29,10,21,1,14,27,8,20,5,17,28,11,23}\}$

Table 3-5: Permutation pattern for the second interleaving

2/ Interleaving depth shorter than the frame

For an interleaving depth shorter than the frame, the first interleaving is not used and the rule of the second interleaving is modified as follows. Let \mathbf{P} be the number of data bits in the interleaving period (\mathbf{P} depends on the modulation used). Let $\mathbf{x(n)}$ and $\mathbf{y(n)}$, for $\mathbf{n = 0}$ to $\mathbf{P-1}$, be the sequences of bits respectively at the input and at the output of the interleaver. Let \mathbf{C} be some positive integer. Let \mathbf{k} be a permutation of the integers from $\mathbf{0}$ to $\mathbf{C-1}$, that assigns a particular integer $\mathbf{k(j)}$ in $[\mathbf{0;C-1}]$ as image of each integer \mathbf{j} in $[\mathbf{0;C-1}]$. The integer \mathbf{P} is expressed by a single manner in the form:

$$\mathbf{P = C \cdot Q - R}$$

where \mathbf{Q} and \mathbf{R} are integers, and \mathbf{R} satisfies the requirement:

$$\mathbf{0 \leq R < C}$$

The set of the integers from $\mathbf{0}$ to $\mathbf{P-1}$ is partitioned into \mathbf{C} consecutive intervals. The interval number \mathbf{j} , for $\mathbf{j = 0}$ to $\mathbf{C-1}$, is:

$$[\mathbf{A(j) ; A(j+1) - 1}]$$

where $\mathbf{A(j)}$ is defined by recurrence:

$$\mathbf{A(j) = \begin{cases} \mathbf{A(j-1) + Q} & \text{if } \mathbf{k(j) < C-R} \\ \mathbf{A(j-1) + Q - 1} & \text{if } \mathbf{k(j) \geq C-R} \end{cases}}$$

with the starting and stopping points:

$$\begin{aligned} \mathbf{A(0)} &= \mathbf{0} \\ \mathbf{A(30)} &= \mathbf{P} \end{aligned}$$

Of course the integer \mathbf{C} has to be chosen $\mathbf{C \leq P}$ in order that any interval $[\mathbf{A(j) ; A(j+1) - 1}]$ contains one integer at least. Then any integer \mathbf{n} , for $\mathbf{n = 0}$ to $\mathbf{P-1}$, belongs to some interval $\mathbf{n^\circ j}$ and to nothing else, and can be expressed by a single manner in the form:

$$\mathbf{n = A(j) + i}$$

where i is an integer that satisfies the requirement:

$$0 \leq i < A(j+1) - A(j)$$

The output bit $y(n)$, for $n = 0$ to $P-1$, is connected to the input sequence by the relation:

$$y(n) = x(C \cdot i + k(j))$$

3.2.2.5 Error correction code

In UTRA standard, two kinds of channel coding schemes are available [3-2]:

- convolutional coding with rate 1/2 or 1/3 and constraint length 9,
- turbocoding with rate 1/3 and an internal block interleaving whose size is 40 or greater.

Turbocoding yields the best performances in bit error rate, but is not well suited to a real time reconfiguration because of its too long constraint length. As we do not intend to reach the best performances, but only to demonstrate the interest of the link adaptation, we adopt the convolutional coding.

This convolutional encoding is defined in the following manner. Let $x(n)$ be the bit sequence at the input of the encoder and $y(n)$ be the bit sequence at the output of the encoder.

For the rate 1/2 code, the coded bits are:

$$y(2n) = x(n) + x(n-2) + x(n-3) + x(n-4) + x(n-8) \text{ modulo } 2$$

$$y(2n+1) = x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-5) + x(n-7) + x(n-8) \text{ modulo } 2$$

with the starting condition:

$$x(i) = 0 \quad \text{for } i < 0$$

For the rate 1/3 code, the coded bits are:

$$y(3n) = x(n) + x(n-2) + x(n-3) + x(n-5) + x(n-6) + x(n-7) + x(n-8) \text{ modulo } 2$$

$$y(3n+1) = x(n) + x(n-1) + x(n-3) + x(n-4) + x(n-7) + x(n-8) \text{ modulo } 2$$

$$y(3n+2) = x(n) + x(n-1) + x(n-2) + x(n-5) + x(n-8) \text{ modulo } 2$$

with the starting condition:

$$x(i) = 0 \quad \text{for } i < 0$$

UTRA FDD standard stipulates to add 8 tail bits with binary value 0 to the end of the data block before encoding.

The ratio of the bit rates when going from a modulation type to the next one (BPSK to QPSK, or QPSK to 16QAM) is 2. For some given modulation, the ratio of the information (before encoding) bit rates when reconfiguring the error correction coding has to be located between 1 and 2. The information bit could be varied almost continuously by applying appropriate puncturing patterns to the coded bit sequence. For the demonstration, the number of error correction codings will be restricted to two different codings per modulation type. The ratio of the information bit rates for these two codings is chosen in the vicinity of 1.5. For example the information ratio with the rates 1/2 and 1/3 convolutional codes is $(1/2) / (1/3) = 3/2$. The same ratio is provided by the rate 1/2 code and the same code punctured by the bit pattern 111001 with a global rate 3/4: $(3/4) / (1/2) = 3/2$. For the sake of simplicity, this last example is selected for the implementation.

From a block of L input bits $x(n)$, for $n = 0$ to $L-1$, an encoding with rate p/q and 8 tail bits yields a block of $q(L+8)/p$ output bits $y(n)$, for $n = 0$ to $\{q(L+8)/p\}-1$. Note that $L+8$ must be a multiple of p .

The number P of bits in a configuration period can be expressed in the form:

$$\mathbf{P} = \mathbf{q} \cdot \mathbf{P}_1 + \mathbf{r}$$

with $\mathbf{0} \leq \mathbf{r} < \mathbf{q}$

In order to fill in the configuration period with the coded bits in the most effective manner, we choose the size \mathbf{L} of the input bits block for satisfying the condition:

$$\mathbf{q}(\mathbf{L} + \mathbf{8}) / \mathbf{p} = \mathbf{P} - \mathbf{r}$$

that is

$$\mathbf{L} = \mathbf{p} \cdot \mathbf{P}_1 - \mathbf{8}$$

and we padd the configuration interval with \mathbf{r} dummy bits. The block size \mathbf{L} and the number of padding bits depend on the modulation type and on the error correction code.

The error correction coding requires an interleaving when the reconfiguration period is not little enough with respect to the variation speed of the propagation channel. But modulation type and error correction code are only known a little time before the start of the configuration interval that uses them. It is not possible to encode an \mathbf{L} -bit block in a small time interval before the interleaving. We could perform in parallel as many encodings with insertion of tail bits at the right positions and as many interleavings as there are possible configuration cases, and select the right output bits when starting the configuration period. But such an implementation is unnecessarily complex. The error correction encoder will be run in a discontinuous manner by loading it at each bit period by a state corresponding to the interleaving rule. That operation can be performed because the constraint length of the convolutional code is short.

For reconfiguration periods short with respect to the propagation channel variation speed, trials will be also carried out without interleaving.

When using the error correction code in a downlink channel with slot format 13 and data blocks that are built as previously described for a slot by slot configuration, the information bit rates are (Table 3-6):

Modulation	Coding rate	Number of information bits	Number of padding bits	Information bit rate kbit/s
BPSK	1/2	27	0	40.5
BPSK	3/4	43	2	64.5
QPSK	1/2	62	0	93.0
QPSK	3/4	97	0	145.5
16QAM	1/2	132	0	198.0
16QAM	3/4	202	0	303.0

Table 3-6: Information bit rates with error correction and slot format 13 (slot by slot reconfiguration)

Those rates are suitable to a video demonstration.

3.2.2.6 Quality parameter estimation

The parameter used for estimating the quality of the transmission channel is very important. It must:

- characterize the transmission accurately enough to be used for controlling the handover between the modulation types,
- not require too long observation time of the received signals for obtaining a sufficient precision (the random component of the parameter decreases when the number of samples that are used for computing increases),
- be simple to be computed in order that the computing time can be shorter than the reconfiguration period,
- allow for maximizing the average information rate that is transmitted for a given output quality.

The parameter that has been chosen for deciding the configuration to be applied is the signal to noise ratio after despreading. It is estimated by remodulating the demodulated symbol and comparing the remodulated symbol with the symbol received at the input of the demodulator. The accuracy of this estimation depends on the number of symbols that are taken into account in the computing and on the stability of the amplitude recovery before demodulation.

From the signal to noise ratio value obtained during some configuration period, the new configuration is decided, and it is applied during the next configuration period. As the propagation channel characteristics have varied from a configuration period to the next one, the new configuration does not always correspond exactly to the quality of the channel. The link adaptation is very effective if the reconfiguration period is short with respect to the fluctuation period of the channel. The experimentation results issued from TRUST have shown that a reconfiguration frame by frame cannot cope with a propagation channel corresponding to a 50 km/h motion and that a reconfiguration on a slot duration or on a few slots is necessary.

In TRUST the signal to noise ratio was estimated by a simple averaging on the power of the noise samples coming from the comparison of received symbols and remodulated symbols. In SCOUT more elaborated filtering methods will be used. These methods will allow to predict the future time evolution of the signal to noise ratio in the limit of half the shortest period of the channel variations. In this way the signal to noise ratio corresponding to a configuration period can be estimated in advance, and the link adaptation can be performed in a better manner. A study will be conducted in order to determine which accuracy can be obtained on the measurement of the signal to noise ratio and on the evaluation of the future time evolution of this ratio.

3.2.2.7 Transmission of the configuration number

There are 3 modulations types and 2 error correction codings, that is 6 possible configurations for transmitting the user data. When the transmission channel provides a very high quality, 16QAM is used and the error correction code can be suppressed. Moreover, when the transmission channel provides insufficient quality at some times, the link adaptation system can decide not to transmit the information signal until the channel quality is good enough again, what is better for the quality of service than to transmit erroneous data. That case is named the "no information transmission" case and plays a prominent part in the link adaptation. Of course, in this case, the radio transmission is not stopped but carries some dummy data in BPSK in order that the estimation of the quality parameter can be executed. For convenience the configuration types are numbered as follows:

- 0: no information transmission (BPSK with padding data)
- 1: BPSK with rate 1/2 error correction code
- 2: BPSK with rate 1/2 error correction code and puncturing
- 3: QPSK with rate 1/2 error correction code
- 4: QPSK with rate 1/2 error correction code and puncturing
- 5: 16QAM with rate 1/2 error correction code
- 6: 16QAM with rate 1/2 error correction code and puncturing
- 7: 16QAM without error correction code

The objective of link adaptation is to guarantee that the Bit Error Rate (BER) cannot pass beyond some fixed value at any time whatever the quality of the propagation channel may be. From the BER curves of each configuration obtained on a Gaussian channel and from the BER that has to be guaranteed, the intervals for using the various configurations in link adaptation mode can be deduced, and the threshold values corresponding to the frontiers of these intervals can be determined. In link adaptation mode, the average BER depends on the characteristics of the propagation channel, but it is always less high than the guaranteed BER.

For representing the configuration number, three bits are necessary. Those bits have to be returned from the receiver to the transmitter. As the hardware validator allows for a one-way transmission only, the configuration number will be returned by a wired connection in a first phase of the experimentation. In reality the configuration number has to be transmitted on the radio link. In a second phase of the experimentation, the configuration number will be inserted into the slot structure near the end of the reconfiguration period in order to be used during the next configuration period. For the sake of simplicity the reconfiguration number will be transmitted on the same radio way as the data, instead of going up a return way, in order that it can undergo the perturbations of the radio transmission. In reality the radio

perturbations would have to be uncorrelated on the return way and on the direct way. That is why the emulation of a return way uncorrelated with the direct way is considered as a possible alternative option.

It is very important to protect the configuration number against the transmission errors. Assuming a propagation channel that provides independent binary errors that occur with a probability p , the probability $P(m,n)$ for m binary errors and over to occur in a packet of n bits is given by the relation:

$$P(m,n) = \sum_{i=m}^n n(n-1)\dots(n-i+1)p^i(1-p)^{n-i} / i!$$

Before error correction, the Bit Error Rate (BER) is very high, e.g. 10^{-3} or 10^{-2} . The probabilities $P(m,n)$ corresponding to packets with short sizes are (Table 3-7):

BER	Packet size (bits)	Number of binary errors:				
		≥ 1	≥ 2	≥ 3	≥ 4	≥ 5
10^{-3}	5	5×10^{-3}	10^{-5}	10^{-8}	5×10^{-12}	10^{-15}
	10	10^{-2}	4.5×10^{-5}	1.2×10^{-7}	2.1×10^{-10}	2.5×10^{-13}
	20	2×10^{-2}	1.9×10^{-4}	1.1×10^{-6}	4.8×10^{-9}	1.5×10^{-11}
10^{-2}	5	5×10^{-2}	10^{-3}	10^{-5}	5×10^{-8}	10^{-10}
	10	10^{-1}	4.5×10^{-3}	1.2×10^{-4}	2.1×10^{-6}	2.5×10^{-8}
	20	2×10^{-1}	1.9×10^{-2}	1.1×10^{-3}	4.8×10^{-5}	1.5×10^{-6}

Table 3-7: Probability of erroneous bits in a packet

Those examples show that the configuration number has to be protected against 4 binary errors. The convolutional code used for the data cannot be used for encoding configuration number and data together because decoding it needs a too long time. A block code (3,17) with minimum distance 9 can be used. The corresponding probabilities for not recovering the right configuration number are:

$$\begin{aligned} 3.2 \times 10^{-12} & \text{ for a BER of } 10^{-3} \\ 3.2 \times 10^{-7} & \text{ for a BER of } 10^{-2} \end{aligned}$$

3.2.2.8 Reconfiguration principles

The validator is controlled by an external PC that loads parameters values and configuration patterns into it. According to the loaded data, the validator performs its configuration itself. In order to avoid some dead period when going from some configuration to another, the validator uses a distributed control with flag signals that configure or reconfigure each part of the validator at the appointed times. Therefore when running in a continuous transmission mode, the validator can be reconfigured without interrupting the transmission. Of course there are some theoretical limitations as constraint lengths (e.g. in coding or interleaving operations) or as block structures (e.g. transmission slots), so that a reconfiguration cannot be carried out at any time, but only at specific accurate times.

The reconfiguration can be started either by the control PC (manual command) or by the validator (e.g. according to the evaluation of a quality parameter inside the receiver). In both cases the reconfiguration is performed by the validator at a time consistent with the transmission system. In the case of the demonstration using an adaptive reconfiguration in a UTRA FDD downlink, when a reconfiguration command is sent to the validator, the parameters of the new configuration are called by the validator, the new configuration is prepared, and when ready the configuration is applied at the end of a configuration interval.

3.3 Video demonstration installation

3.3.1 Description of the interaction with the video data flow

The video encoder supplies the hardware validator with successive video packets. The validator is running with adaptive link adaptation in a UTRA FDD downlink as previously described.

The validator and the video system act as fully separated systems. The video source and video decoder operate on two different networked PC systems. The validator and the video system are connected together by a TCP link. The main video data path is unidirectional: video source-validator-video decoder. The transmit part of the validator generates a data rate in accordance to the configuration case and to the UMTS frame format that are in effect.

The video system can work without any control or information from the validator because the data flow is regulated by the TCP link between video source and validator: the average bit rate of the video data is kept less or equal to the rate of the validator.

3.3.2 Transmission of the video packets

The hardware validator runs in a continuous transmission manner. The video packets have to be inserted in a continuous data flow. A training word will be added to each video packet, in order to identify and extract the video packet from the received data. The training word will be distributed over all the video packet, so that no particular data pattern can be confused with it. An objective is that the insertion of the training word introduces an overrate lower than 5% for video packets long enough.

The training word and the video data undergo a convolutional coding. At the reception after decoding, they are recovered with a bit error rate that is wanted lower than 10^{-6} by the link adaptation mechanism. Moreover the errors after error correction decoding are often supplied as bursts of errors. Therefore the bits of the training word have to be split over the data with gaps larger than the errors packet sizes. Assuming independent binary errors that occur with a probability p , the probability $P(m,n)$ for m binary errors and over to occur in a packet of n bits is given by the relation:

$$P(m,n) = \sum_{i=m}^n n(n-1)\dots(n-i+1)p^i(1-p)^{n-i} / i!$$

The probability $P(m,n)$ corresponding to a training word with a size around a hundred bits is (Table 3-8):

BER	Word size (bits)	Number of binary errors:	
		≥ 1	≥ 2
10^{-6}	50	5×10^{-5}	1.2×10^{-9}
	100	10^{-4}	5×10^{-9}
	200	2×10^{-4}	2×10^{-8}
10^{-7}	50	5×10^{-6}	1.2×10^{-11}
	100	10^{-5}	5×10^{-11}
	200	2×10^{-5}	2×10^{-10}

Table 3-8: Probability of erroneous bits in a packet

That table shows that the training word has to withstand one erroneous bit.

Each video packet starts with a 128-bit header, that contains very important informations as the length of the packet. By using a block code (4,7) with minimum distance 3, the header data are transmitted by 32 coded words of 7 bits allowing for a protection against one error in each word. the probability for having some error in the header after decoding is (Table 3-9):

BER before decoding	Probability of erroneous decoding of a coded word	Probability of erroneous decoding of the header
10^6	2.1×10^{-11}	6.1×10^{-9}
10^{-7}	2.1×10^{-13}	6.7×10^{-11}

Table 3-9: Probability of erroneous decoding of the header

These values, obtained for the header, are consistent with the protection of the training word. practically the bits of the training word and the coded bits of the header will be interleaved in order to cope with the error bursts.

When the video source does not provide data to the validator, the validator adds some dummy data after the last transmitted video packet. At the reception the video packet length is extracted from the header to determine the end of the video packet, and to discard the dummy data if any.

3.3.3 Quality of transmission

The header of each video packet contains a value of the requested maximum bit error rate on the transmission of this packet. In a first time of the experimentation, the hardware validator will not take this value into account and will transmit all the video packets with the same quality of transmission. In a second time, two levels of quality will be defined: ordinary quality and improved quality. The comparison of the requested maximum bit error rate with some predefined threshold value allows for selecting the quality level in the transmitter. As there is no direct relation between the video packet lengths and the reconfiguration intervals, a video packet is not necessarily transmitted with the requested quality: when a video packet of ordinary quality and a video packet of improved quality are transmitted during the same configuration period, the improved quality is used for the both. If the validator ends the transmission of a video packet in ordinary quality, and if a video packet appears for improved quality transmission, the validator transmits dummy data until the next reconfiguration.

The quality level (ordinary, improved) has to be supplied by the transmitter to the receiver. In a first time the quality level will be transmitted by a wired connection. In a second time of the demonstration, it will be inserted in the slot structure by using a method similar to the return of the configuration number from the receiver to the transmitter, with a protection against the radio transmission errors.

For implementing those two different quality levels, two ways are possible. The first one consists in selecting the configuration the most appropriate among the already existing configuration cases (modulation, error correction code). The number returned by the receiver to the transmitter is considered as defining a transmission quality interval. From this information and from the requested quality level, the transmitter chooses a configuration number. In this case the transmitter sends the whole configuration number, not the quality level indicator, to the receiver.

The second way consists in doing a selection between different error correction codes, without changing the modulation. For example the convolutional code with rate 1/2 is used in ordinary quality, and the code with rate 1/3 in improved quality. In this case, the receiver needs only to know the quality level to be used.

A third way would consist in using concatenated codes: the convolutional code already implemented for the link adaptation demonstration and an external code, e.g. a Reed Solomon code, with reconfigurable capacity of correction. But such a solution needs a long interleaving between the two codes, and would introduce an important delay on the video transmission. That is why it is rejected.

3.4 Conditions of the experimentation

3.4.1 Demonstration system

The overall demonstration system is composed of (**Figure 3-1**):

- the video system (video source and video decoder),
- the validator (transmitter, receiver, control and display system),

- the simulated propagation channel (hardware channel simulator, noise generator).

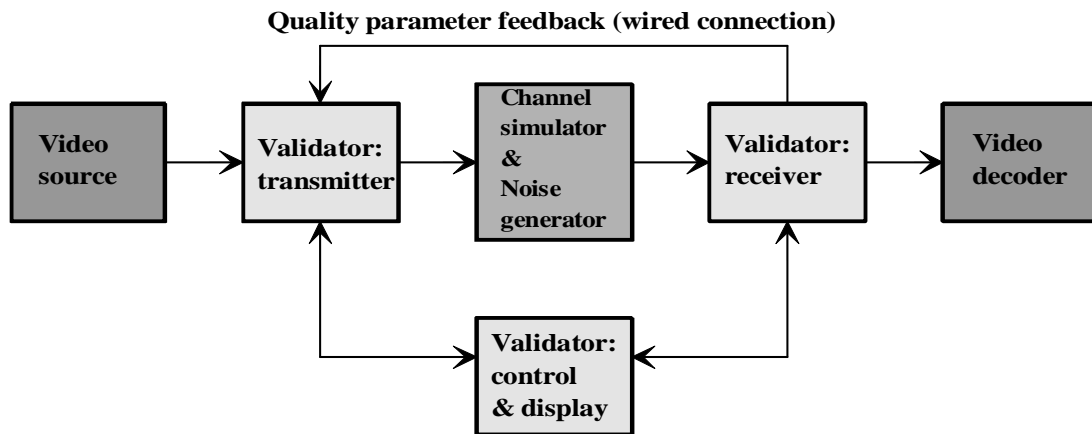


Figure 3-1: Demonstration system

As the hardware validator performs a one-way transmission only, the informations that are necessary to control the link adaptation, i.e. the measurement results of the quality parameter, are sent from the receiver to the transmitter by a wired connection.

3.4.2 Control/display environment

3.4.2.1 Control

The control of the hardware validator and the display of the measurement results is carried out by a PC. The control/display operations to be performed are:

- the initial configuration of the programmable devices inside the validator,
- the loading of parameter values or of specific data into the validator,
- the extraction of some data elaborated by the validator,
- the display of the transmission environment and of the results.

The user data flow that is transmitted and received by the hardware validator does not go through the control PC; the control PC can only regulate its transmission rate. The processings that need a large amount of computing power (e.g. estimation of the quality parameter) are performed inside the hardware validator. The simplest statistical operations (e.g. average value of the information rate) are performed by the control PC.

When starting the demonstration, the control PC sends all the transmission parameters values and the running mode for storage into the hardware validator. At any time, from the keyboard of the PC, a parameter value or the running mode can be changed. In automatic handover mode, the hardware validator performs the real time reconfiguration without any help from the control PC.

3.4.2.2 Transmission control parameters

The parameters that define the main characteristics of the transmission are:

- filtering coefficients,
- carrier frequency with respect to the centre frequency of the frequency band processed by the validator,
- modulation type,
- symbol rate,
- transmit power control,
- spreading factor and slot format,
- spreading sequence number,
- scrambling sequence number.

In the case of forward error correction, there are also:

- error correction code,
- depth of the associated interleaving.

For the link adaptation, we must add the parameters that follow:

- time window for computing the quality parameter that allows for deciding which configuration is to be used,
- threshold values for switching between the configurations.

For the reconfiguration demonstration, the most important control parameters are the configuration number and symbol rate, the command for computing the quality parameter, and the decision thresholds.

3.4.2.3 Measurements

The measurements to be performed are:

- the signal to noise ratio, that is used as quality parameter for controlling the configuration changes,
- the bit error rate,
- the relative frequency of the handovers with respect to the frame frequency,
- the average information rate,
- the distribution of the propagation paths that are detected when despreading and descrambling the CDMA received signals.

All those measurements have to be controlled by some parameters:

- error counting duration,
- observation times for the statistical measurements (configuration changes, average rate...).

The measurement parameters (e.g. error counting duration) will be entered from the keyboard of the control PC.

3.4.2.4 Display

Some part of the screen of the control PC will be assigned to display some important parameters or results in a permanent manner. They are:

- the main transmission parameters: modulation type, error correction code, data rate...
- the constellation of the modulation states,
- the main results: bit error rate, average information rate, statistical frequency of reconfiguration...

The other parameters will be displayed when asked from the keyboard of the control PC.

Another part of the screen will be used for specific graphical displays. The possible displays are:

- signal to noise ratio, modulation, and instantaneous bit rate versus time,
- bit error rate versus **Es/No** ratio,
- propagation paths versus delay.

3.4.3 Phases of the demonstration

The experimentation will start by measuring the general transmission performances of the validator: bit error rate versus **Es/No** ratio of the energy per symbol **Es** to noise power density **No**. Then trials will take place in order to validate the real time reconfiguration scenario: the link adaptation in a UTRA FDD downlink. Finally the scalable video algorithms will be used on the hardware validator system and their interaction with the link adaptation will be investigated.

3.4.3.1 Transmission performances measurement

The first step of the experimentation is the measurement of the general transmission performances of the validator, in order to verify that the validator runs in a correct manner and to make sure that the results of the further trials can be valid and realistic. These measurements will be made by transmitting a pseudo

random binary sequence through the validator. The trials have to be carried out for the various modulation and error correction types and for several spreading factors.

On a Gaussian channel, the bit error rate versus the E_s/N_0 ratio will be measured, and the obtained performances will be compared to the theoretical performances of the BPSK, QPSK and 16QAM modulations with and without error correction coding. On a multipath propagation channel, the trials will consist in verifying that the propagation paths are detected and in measuring the bit error rate versus E_s/N_0 .

The characteristics of the real propagation channels are extremely various. For the demonstration, the "Vehicular Test Environment- Channel A" model from ITU-R [3-4] will be used. The characteristics of this ITU-R channel are (Table 3-10):

Path number	Delay τ (μ s)	Relative power (dB)	Doppler
0	0.00	0	CLASSIC
1	0.31	-1	CLASSIC
2	0.71	-9	CLASSIC
3	1.09	-10	CLASSIC
4	1.73	-15	CLASSIC
5	2.51	-20	CLASSIC

Table 3-10: ITU-R Vehicular Test Environment - Channel A

In that table "CLASSIC" means that the power spectrum density $S_i(f)$ of the path number i is:

$$S_i(f) = \begin{cases} A_i \{ 1 - (f / fd) \}^{-1/2} & \text{for } |f| < fd \\ 0 & \text{for } |f| \geq fd \end{cases}$$

where A_i is a constant coefficient and fd is the Doppler frequency.

The reference will always be the bit error rate versus E_s/N_0 on a Gaussian channel.

3.4.3.2 Link adaptation test

The aim of the link adaptation experimentation is:

- to demonstrate the actuality of a real time reconfiguration,
- to validate the performances improvement effected by the link adaptation, i.e. the increase of the average information rate that can be transmitted for a given quality.

The functions that are involved in the reconfiguration process are:

- in the transmitter: BPSK/QPSK/16QAM modulation, interleaving, error correction encoding, bit rate generation;
- in the receiver: coherent BPSK/QPSK/16QAM demodulation, rate regeneration, deinterleaving synchronization, error correction decoding.

The spreading and scrambling sequences generation, the filtering, the frame synchronization, the despreading and descrambling are programmable operations, but they are common to all the configuration cases. The reconfiguration will be controlled by a flag signal that will propagate from function to function in order to switch the parameters of each function at the right time. That flag signal will be under control of a chip or sample counter synchronized on the configuration interval. The decision thresholds to be used for the link adaptation are deduced from the measurement results obtained on a Gaussian channel.

The demonstration will be led in successive phases.

1st phase

The reconfiguration is performed automatically by using the quality signal that is computed inside the receiver. The receiver returns the configuration number to the transmitter by a wired connection.

2nd phase

The control part of the DPCH channel contains a TFCI field. Some bits of that field will be used for transmitting the configuration number, in order to simulate a return way. Optionally a return way with channel fluctuations not correlated with the fluctuations of the direct way will be implemented.

3rd phase

The estimation of the signal to noise ratio is performed in the current configuration time interval and applied in the next configuration interval. When the channel fluctuates quickly, the performances of the link adaptation are reduced. In order to improve the adaptation, some methods will be developed for estimating the future evolution of the channel.

In all these phases the demonstration will consist in measuring the bit error rate and the average bit rate and in comparing them with the bit error rate and bit rate in QPSK, in order to determine the gain on the link budget when using link adaptation.

3.4.3.3 Video demonstration phases

The video demonstration will be conducted in two successive phases.

1st phase

The hardware validator does not take into account the quality requirement located in the header of the video packets. A propagation channel with one or several paths and with noise is used. First a subjective evaluation of the video algorithms will be carried out for each static configuration case. Then the interaction of the scalable video algorithms with the link adaptation will be studied.

2nd phase

From the quality requirement of each video packet, the hardware validator selects one level of quality (ordinary or improved) and applies the appropriate configuration, according to the current quality of the propagation channel. The improvement of quality on the video with respect to the uniform quality will be studied.

3.5 Conclusion: benefits of the link adaptation

The experimentation that has been carried out during TRUST project has shown that the adaptive modulation supplies a high gain onto the link budget for radio link on multipath propagation channels with Doppler. When using adaptive modulation and adaptive error correction coding, it is expected to obtain a higher gain. The experimentation to be led in SCOUT project will allow to determine the conditions of a realistic use of the link adaptation. Moreover the interaction of the link adaptation with the scalable video transmission will be studied in order to see what sort of improvement can be provided.

3.6 References

- [3-1] 3GPP TSGRAN: Physical channels and mapping of transport channels onto physical channels (FDD)
(3GPP TS25.211 v4.1.0 2001-06)
- [3-2] 3GPP TSGRAN: Multiplexing and channel coding (FDD)
(3GPP TS25.212 v4.1.0 2001-06)
- [3-3] 3GPP TSGRAN: Spreading and modulation (FDD)

(3GPP TS25.213 v4.1.0 2001-06)

[3-4] ITU-R M.1225: Guidelines for Evaluation of Radio Transmission Technologies for IMT-2000
(Question ITU-R 39/8 1997-02)

[3-5] TRUST Deliverable D2.4: Radio Transmission demonstration conclusions: experimentation results
analysis, performance evaluation
(IST-1999-12070 TRUST D2.4, January 2003)

4. Video Demonstration

4.1 Introduction

This section covers the work of A4.3.4, A4.3.5, and the interactions with A4.3.3 within the SCOUT project. A4.3.4 and A4.3.5 are primarily concerned with the use and development of scalable video coding systems for use with reconfigurable wireless systems. The SCOUT project will demonstrate these technologies in conjunction with the SCOUT hardware validator developed in A4.3.3.

Section 4.2 provides a brief review of current scalable video systems. Section 4.3 considers the application scenarios in which scalable video systems would be used. Section 4.4 considers the relevant IP protocols and traffic control systems which would be found in a complete system. Section 4.5 gives an overview of the SCOUT video demonstration system, while sections 4.6 and 4.7 define the structure and software interfaces that will be used. Section 4.8 briefly summarises the objectives of the video demonstration.

4.2 Current Scalable Video Systems

A scalable video bit stream is one, which can be decoded at different rates, according to the bandwidth available for transmission to the decoder. This enables a user with access to a higher bandwidth channel to decode high quality video, while a lower bandwidth user is still able to view the same video, albeit at a lower quality. Consequently a scalable video bit-stream inherently contains data of differing importance or priority. A reconfigurable radio system may be able to offer a better quality of service to higher priority video information than lower priority information. Consequently during periods of poor reception, the system may transmit just higher priority information while periods of good reception would allow both high and low priority data to be transmitted, thereby resulting in improved visual quality.

This section provides a brief summary of the different mechanisms or types of scalability which exist in current video coding systems including H.263+ [H.263+] and MPEG-4 [MPEG-4].

4.2.1 Overview of H.263+

The ITU-T standard H.263+ [H.263+], also called H.263 version 2, is an extension of the original H.263 standard incorporating extra annexes, the most significant of which for scalable video coding is annex O. H.263+ is a layered scalable coding scheme. This means that the bit stream is organized in a hierarchical structure, divided into a number of levels, the base layer and one or more enhancement layers. The base layer, encoded at a low bit rate, is the minimum needed to decode the video sequence, and consists of an intra coded picture (known as an I picture) and multiple inter-coded pictures (known as predicted or P pictures). The enhancement layers, which require additional bandwidth for encoding, improve upon the spatial or temporal quality of the reconstructed base layer sequence. Thus, an enhancement layer can only be decoded if the layer immediately below it, known as the reference layer, has already been decoded. Annex O defines three different types of scalability: temporal, spatial and SNR scalability.

4.2.1.1 Temporal scalability

Temporal scalability increases the frame rate of the encoded sequence by inserting additional frames into the sequence, which are encoded by bi-directional prediction. That is, they are predicted from the previous and subsequent P picture, so that they are disposable in nature.

Figure 4-1 shows an example of temporal scalability. The base layer typically consists of a combination of I-frames (intra-frame coded) and P-frames (coded using motion compensated prediction). The enhancement layer consists of B-frames (bi-directionally predicted from previous and subsequent frames).

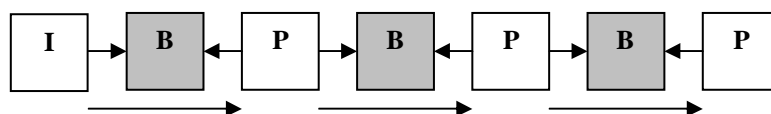


Figure 4-1: Temporal scalability. The un-shaded portions represent the base layer while the shaded parts represent the enhancement layer.

A low delay version of temporal scalability can be realized by using intervening P frames (relying on forward prediction only).

4.2.1.2 SNR scalability

SNR scalability refers to the process of improving the signal to noise ratio of the base layer picture by including additional information in the enhancement layer. This additional information is encoded as either EI (Enhancement Intra) or EP (Enhancement Predicted) pictures. An EI-picture is only upward predicted from a decoded picture in the reference layer and does not use motion vectors, whereas an EP-picture can be either upward predicted from the reference layer or forward predicted from a previous picture in the enhancement layer, or use some combination of both prediction modes. When EP-pictures are forward predicted from a same-layer enhancement picture, motion vectors are required. Figure 4-2 shows an example of SNR scalability comprising a base layer and 2 enhancement layers. EP frames can offer greater compression for sequences with large temporal correlations, including sequences with stationary regions. However, the use of EP frames can lead to drift problems when a system switches up from base-layer only to base and enhancement layer operation. Here the term drift refers to the temporal propagation of missing data to subsequent EP frames.

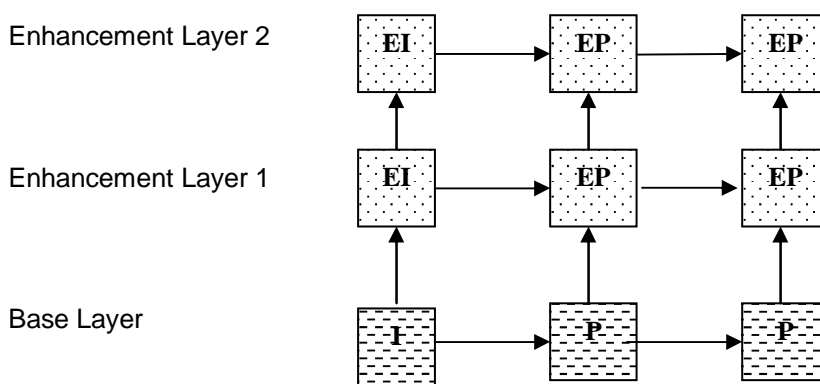


Figure 4-2: SNR and/or spatial scalability.

4.2.1.3 Spatial scalability

Spatial scalability uses the additional bandwidth provided in the enhancement layer to increase the spatial resolution of each base layer picture. Figure 4-3 shows an example of spatial scalability where the base layer is coded at QCIF resolution (176x144) and the enhancement layer is coded at CIF resolution (352x288).

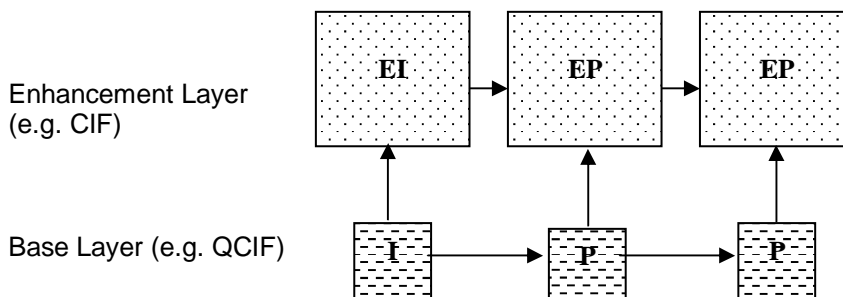


Figure 4-3: Spatial scalability

4.2.1.4 Hybrid scalability

The term hybrid scalability refers to the use of two or more of these scalability modes. Figure 4-4 shows an example of hybrid scalability comprising a base layer, an SNR scalable enhancement layer and a temporal scalable 2nd enhancement layer.

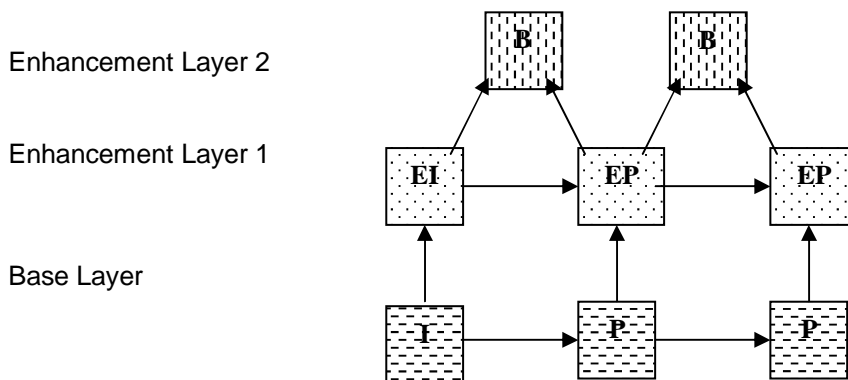


Figure 4-4: A possible configuration of pictures for hybrid scalability

4.2.1.5 Efficiency of H.263+ layered scalability

A layered video bit-stream can only be decoded at the specific rates at which the layers were encoded. More flexibility is gained by having a large number of layers, (the H.263+ standard allows for up to 15 scalable layers). However, the use of more layers typically reduces compression performance significantly. This is a result of both additional headers for each layer, and some repetition of data between different layers. In particular a lot of positional information needs to be included in each layer, which can be efficiently combined in a single layer system. Therefore, in order to maintain good compression efficiency, the number of layers should be kept low (e.g. 2 to 4).

4.2.2 MPEG-2 Scalability

MPEG-2 [MPEG2] is a generic video and audio codec that can be applied to many applications such as HDTV, DVD, and video conferencing. Depending on the profile used, MPEG-2 provides coding at compressed bit rates from less than 4Mbps up to bit rates of 80 Mbps. MPEG-2 includes similar scalability options as H.263+ including temporal, spatial and SNR scalability. One technique which MPEG-2 does have is data partitioning. This is a mechanism for splitting a single layered bit-stream into two portions with different priorities (high and low). The intention is that both layers would be transmitted, however the decoder will suffer less from errors in the low priority data than in the high priority data. Thus the low priority data can be transmitted with a less reliable technique or lower quality of service (QoS). It should be noted that this mechanism is intended for error resilience purposes rather than scalability since the decoder will not perform well given just the base layer.

4.2.3 MPEG-4 Scalability

The MPEG-4 standard aims to provide tools and algorithms for efficient storage, transmission and manipulation of video data in multimedia environments, bringing together the fields of digital television, interactive graphics applications and interactive multimedia such as can be interfaced via the world wide web. MPEG-4 aims to provide better compression efficiency and universal access over a wide range of media. The MPEG-4 standard has been explicitly optimized for three bit-rate ranges: below 64 kbit/s, 64 to 384 kbit/s and 384 to 4 Mbit/s. The aim of MPEG-4 is to cater for the needs and requirements of a wide spectrum of multimedia applications, from digital television and streaming video to mobile multimedia and games.

A key difference between MPEG-4 and previous video coding standards is that MPEG-4 is designed to work with arbitrarily shaped video objects rather than just complete video frames. This enables content-based applications. MPEG-4 also supports a variety of scalability options. These include temporal, SNR, spatial and hybrid layered scalability options similar to those of H.263+ discussed above. In addition, MPEG-4 also enables object-based scalability and fine-grained scalability.

4.2.3.1 Object based scalability

Since MPEG-4 allows the video to be described as a collection of arbitrary shaped objects, scalability options can be applied to any object within the frame. Figure 4-5 shows an example of object based

temporal scalability (although object based scalability can equally be applied to spatial or SNR scalability). Since the scene to be coded is divided into a number of distinct video objects, such as the foreground object and background, and each object is independently coded, the system could choose which video objects to transmit and decode depending on the available bandwidth, end terminal computing power and end user preferences. The number of available layers and hence the number of different bit-rates at which the bit-stream can be decoded is equal to the number of video objects in the scene.

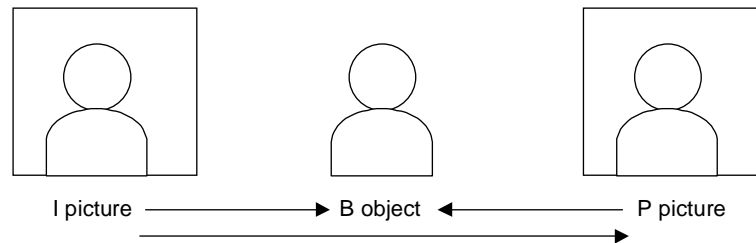


Figure 4-5: Object based temporal scalability

4.2.3.2 Fine granular scalability (FGS)

Layered or coarse-grained scalable video can only be decoded at the specific bit-rates at which the layers were encoded. A major problem with this approach is that compression efficiency usually decreases as the number of layers is increased. Thus if compression efficiency is to be maintained the number of layers must be kept low. As a result, when video is transmitted over a heterogeneous network with dynamically varying bandwidth, the video quality will not vary continuously but in discrete steps between layers.

In order to address the problem of video transmission over heterogeneous networks, a technique known as Fine Granularity Scalability (FGS) has been developed for MPEG-4. FGS enables an encoded video sequence to be decoded at any bit-rate less than the encoded bit-rate on a frame by frame basis. This means that the scaling function can choose to filter out a different number of bits for each frame, depending on the available bandwidth at the time the relevant packets pass through the scaling function. This is achieved by first coding a base layer using conventional non-scalable MPEG-4 encoding. The difference between the original picture and the coded base layer picture is then found. This difference is then coded using bit-plane coding of the DCT residuals to yield the enhancement layer. No motion compensation is allowed in the enhancement layer pictures, that is, EP type pictures are not allowed, only EI pictures are. The result is that the enhancement layer bitstream is continuously (or fine grain) scalable on a frame basis.

Motion compensated prediction must be carried out on the base layer only in order to avoid drift when the enhancement layer is not completely decoded by the decoder. The main drawback is that the lack of motion compensation in the enhancement layer means that the temporal redundancy cannot be exploited in enhancement layer. This can restrict the coding efficiency considerably when compared with single layer coding.

FGS is similar to SNR scalability, but with no restrictions on the bit-rate of the decoded enhancement layer. FGS has also been extended to include temporal scalability.

4.2.4 Future Scalable Video Systems

The current generation of scalable video systems offers a range of different forms of scalability discussed above including both layered or coarse-grained and fine-grained techniques. The choice of scalability mode to employ is made at the encoder during the encoding process. Thus during transmission, the system can merely select which layers to transmit, and in the case of FGS how much of the FGS layer to transmit. While this offers a degree of flexibility in terms of selection of the final content, there are likely to be situations in which a greater degree of flexibility is desirable.

Future systems may offer a greater scope in terms of mechanisms of scalability. It is envisaged that a system may offer a greater variety in terms of the number of different layers, different varieties of fine-grained scalability, and different types of scalability including:

- **Region of interest scalability.** Here a region of the picture may be highlighted as more important and thus given greater priority. Ideally this region may be user dependent, for example it could follow the viewers eye movements or foveation region. Since this region may not be known at the encoder, it is desirable for the bit-stream to support any region

- **Luminance/chrominance scalability.** This mechanism would allow the luminance information to be given higher priority than the chrominance information.

4.2.5 Challenges in future scalable systems

One of the main problems encountered in current scalable systems is the inability fully exploit redundancy between different layers. This can result in a significant loss of compression efficiency. A second major problem encountered is that of drift or temporal error propagation which precludes a codec from fully exploiting the temporal correlations between data in lower priority and FGS enhancement layers. This also results in a significant loss of compression efficiency. These topics are the main focus of work within SCOUT A4.3.5.

4.3 Scalable Video Application Scenarios

Video applications can be divided according to the requirements for real-time and multiple recipients. This division is summarized in Figure 4-6.

	Unicast	Multi-cast
Non-real time	Video on demand	Video broadcasting
Near real time	Video surveillance	Live video broadcasts
Real time (bi-directional)	Video-phone Tele-medicine	Video conferencing

Figure 4-6: Example video applications categorized according to real-time and multi-cast constraints. The shaded region represents the main areas where scalable systems are applicable. The SCOUT video demonstration will focus on the non-real time uni-cast case.

For real time (and to a certain extent near real time) uni-cast applications, it is possible for the encoder to dynamically adapt. In this situation the video encoder can be responsible for maximizing the users perceived quality of service for the given networking conditions. Thus a single layer non-scalable solution can be efficiently used. Note that this relies on the existence of a reverse channel for relaying networking conditions back to the encoder as shown in Figure 4-7.

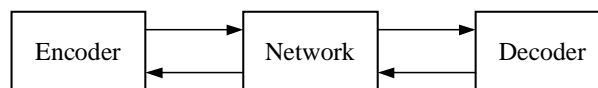


Figure 4-7: Block diagram of a typical real time bi-directional system.

However, for non real time (download) applications, the encoder can no longer adapt to the unknown networking conditions that will be encountered. Instead the encoder needs to generate a scalable bit-stream which can be later scaled to suit the given networking conditions, as shown in Figure 4-8.

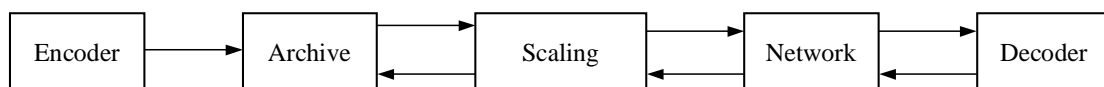


Figure 4-8: Block diagram of a typical non-real time (download) system.

For multicast systems the encoder needs to be able to support a variety of different users with different connectivities. This can be done by providing a scalable video bit-stream, which is then scaled appropriately for different users, as shown in Figure 4-9. Here either of the networks may comprise a combination of wired and or wireless components. There is a similar choice as to where to perform the scaling.

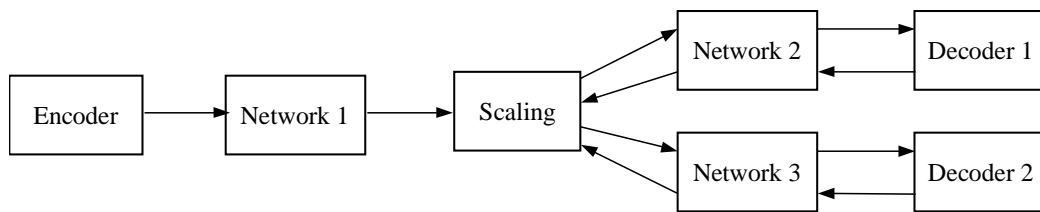


Figure 4-9: Multicast Distribution over Different Networking Systems.

With multicast systems, there are also a variety of scenarios depending on whether multiple recipients are co-located on the same wireless link (e.g. in the same meeting room) or on different locations.

A typical system might comprise a video source connected over a wired network to a fixed base station and then over a wireless network to an end user terminal as shown in Figure 4-10 and Figure 4-11. Here we see that the scaling could be performed at one of 2 locations, either the source and/or the edge router. The advantage of performing the scaling at the source is both a reduction of unnecessary traffic over the wired network, while scaling at the edge router may offer quicker response to changing conditions on the wireless network.

There is also a choice as to whether scaling is performed at the network layer (Figure 4-10) or the application layer (Figure 4-11). The advantage of scaling at the application layer is a greater flexibility over the type of processing performed. However, this may come at the expense of extra associated delays, and the difficulties/protocols involved in running the associated software on a edge router which may have limited resources and heavy loading. Another issue involved in application layer scaling is to ensure that the scaling process has the necessary information about the currently available network services (bandwidth and delay options etc.) Scaling at the network layer avoids the need for additional application scaling software in the edge router. However, network scaling is limited by to the supported traffic control mechanisms. In particular this may not allow the truncation of packets, which would be of use to fine-grained scalable systems.

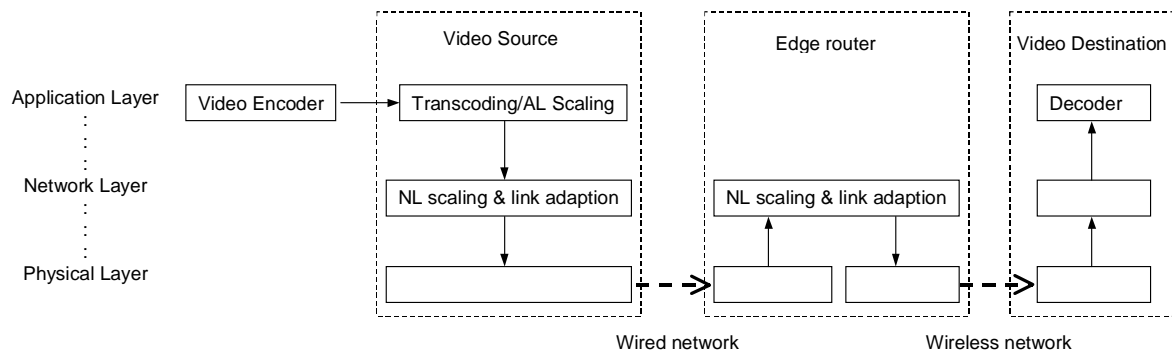


Figure 4-10: Typical system comprising a combination of wired and wireless networks, with scaling at the network layer.¹

¹ Note that a complete system will also contain a return path. This has been omitted for clarity.

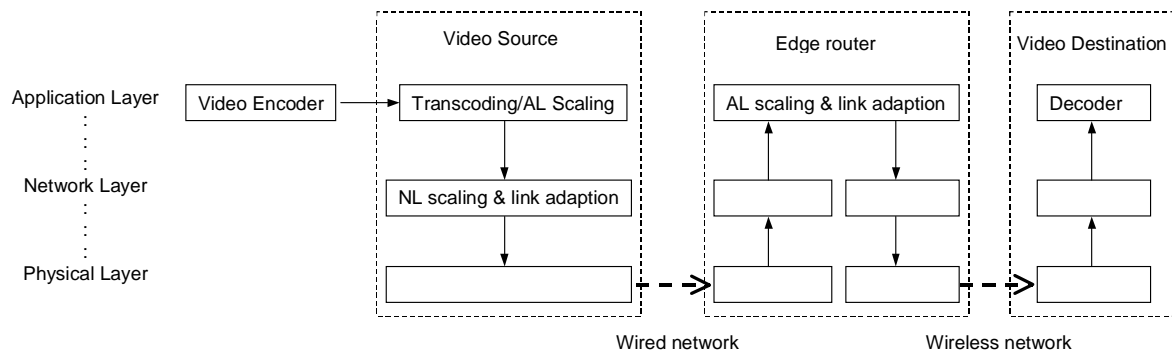


Figure 4-11: Typical system comprising a combination of wired and wireless networks, with scaling at the application layer.¹

4.4 IP Protocols

A typical video communications application can be viewed as a layered system (according to the ISO reference model), with the physical layer at the bottom and the application layer at the top. Between these layers will be various other layers including transport layer, network layer and link layer. One of the most widely used forms of network layer is the internet protocol which provides a connectionless best effort packet system.

The recent growth and convergence between telecommunications and computing industries has seen a rapid growth of the internet and the wide-spread adoption of the internet protocols (IP) over a wide range of communications systems. To this end it is likely that many future systems will include IP as a networking layer on top of a wide variety of underlying physical layer systems. Thus it is likely that many application layer systems including video will reside on top of an IP networking layer. For this reason it is important to understand how the application layer may interact with the IP networking layer. The IP protocols developed by the Internet Engineering Task Force (IETF) [IETF] cover a wide range of applications. In this section we provide a brief review of the most relevant protocols and traffic control mechanisms for transmission of real time scalable video.

4.4.1 Related IP protocols

This section provides a brief review of the relevant IP protocols, which may be used, for the transmission of video data.

4.4.1.1 IP-V4, IP-V6

The Internet protocol (IP) is the most fundamental part of the IP protocol suite. It provides a common network layer protocol which is based on a best effort transmission of IP packets. Note that the network does not guarantee:

- That all or even any packets are delivered.
- Any limit on the transmission delay suffered by packets.
- That packets are delivered in order.
- Packets are not repeated (multiple copies received).

There are currently two relevant versions of IP.

- IP version 4 (IPv4) [RFC791] is in common use in most current routers and systems.
- IP version 6 (IPv6) [RFC2460] is more recent version which is designed primarily to combat the rapid growth of the internet both in terms of providing larger addresses and larger routing tables.

Each IP packet includes both a source and destination address (32bit each for IPv4 and 128 bits each for IPv6). IP supports both uni-cast one-one communications and multi-cast one-many broadcasting of packets. Multicast packets can be determined from the IP address. The IP addresses are used by the network to route the packet through the network to the end user. This routing is typically done independently for each packet. The IP addresses can also be used by traffic control mechanisms to classify the packet for different services.

IP in itself is intended to be used in conjunction with an appropriate transport layer protocol such as TCP or UDP.

4.4.1.2 Transport protocols – TCP and UDP

Most of the information conveyed between applications in an IP network is encapsulated into one of two transport protocols (UDP or TCP). In order to distinguish the target application, both protocols also include two 16 bit port numbers for both the source and destination devices. These port numbers can also be used by traffic control mechanisms to help classify the packet for different services.

The transmission control protocol (TCP) is a transport layer protocol intended to provide reliable stream or connection orientated service. The aim is to provide a reliable transparent pipe between applications. TCP is commonly used for many applications, which rely on reliable transfer without delay constraints. Applications include file transfer. TCP achieves reliability by utilizing IP in conjunction with ARQ techniques to retransmit lost packets. Although TCP provides a reliable service, it is not suitable for real time video streaming applications over a large or limited bandwidth network.

The user data-gram protocol (UDP) provides a transport protocol, which provides a more transparent access to IP. Thus UDP offers the same features as IP in terms of a best effort packet based service with no reliability guarantees. UDP is well suited to both multi-cast (broadcast applications) and also to applications where lower latency is more important than data integrity. Thus UDP is well suited to real time services such as audio or video which should be able to recover from occasional packet loss. Although UDP can be used directly to provide these types of services, a more specific protocol exists which provides a range of additional features.

4.4.1.3 Real time protocol (RTP)

The real time protocol (RTP) [RFC1889] is intended for the transmission of real time data across a packet based network. Although the underlying transport or networking layer is not specified, it is common to use RTP over UDP and IP. RTP actually encompasses an additional protocol, namely the real time control protocol (RTCP) which is used to manage RTP sessions. RTCP allows the use of both uni-cast and multicast sessions, and allows users to join and leave a session. Each RTP session is intended to contain only one type of content. Thus an audio-video conference will typically comprise at least two sessions (per transmitting participant) for audio and video data respectively. This allows each receiving participant to select which sessions he wishes to receive. Thus a participant can select audio only or both (or even video only).

Within an RTP session, no distinction is made as to the content of different packets, although priority distinction could be made using another technique. Thus in order for RTP to be used with a scalable video system, each layer typically needs to be encapsulated in a different RTP session. A participant can select as many sessions as required. However if this selection is to be made adaptively in response to fluctuating networking conditions, then the system may result in many changes in terms of the streams selected. This is likely to cause larger overheads in terms of the RTCP traffic.

A number of standards also exist for the encapsulation of common content types/standards including: MPEG-1 & MPEG-2 [RFC2250], H.263 [RFC2190], H.263+ [RFC2429] and MPEG-4 [RFC3016]. The scalability options are supported by the use of multiple RTP sessions. As yet no support is made for fine-grained scalability (FGS), without requiring an application layer scaling agent.

RTP and RTCP provide the necessary protocols for transmitting and managing the transmission of real time protocols. However since they rely on an underlying unreliable packet based network such as that provided by UDP/IP, they do not make any guarantee of the quality of service.

Other relevant protocols include the Real Time Streaming Protocol [RFC2326] and the multimedia gateway "Megaco" protocol [RFC3015]. These are not discussed further since they describe higher levels of service control.

4.4.2 Traffic Control Techniques

This section provides a brief review of the different methods, which can be used, for traffic control within an IP network. Traffic control refers to techniques for specifying different behavior for different packets or streams of packets. This includes both mechanisms for varying priorities and constraints of delay, throughput, reliability and cost. Essentially they allow network nodes to treat different packets differently according to the relevant fields determined from the packet header. This is of particular relevance to scalable video where we may wish to assign different priorities to different layers or portions of the encoded bit-stream.

4.4.2.1 IPv4 relative priority marking

The original version of IPv4 [RFC791] specified a type of service (TOS) field (8 bits) comprising 3 bits precedence and one bit each to specify: normal or low delay, normal or high throughput and normal or high reliability. The remaining 2 bits are unused and set to 0. This technique offers 8 distinct priority levels, of which several are pre allocated for network control functions. Although the original intention was to provide variation in prioritization, few networks or routers supported these features.

4.4.2.2 IPv4 type of service (TOS)

The meaning of the IPv4 type of service (TOS) octet was updated/clarified in RFC1349 [RFC1349]. Here the 8 bits are split into 3 bits precedence (as described above), 4 bits TOS and 1 unused (zero) bit. The 4 bit TOS field allows a total of 16 different types of service. Five of these are given special meaning including: default behavior, minimize delay, maximize throughput, maximize reliability and minimize cost.

The problem here is that the flexibility here is still largely limited and still offer only a very limited set of precedence with no guarantee as to how these are treated, other than guaranteeing that the higher precedence or no-default TOS packets will not suffer an inferior service than default packets.

In IPv6 [RFC2460], the 8 bit traffic class can be used in the same way as the TOS field.

4.4.2.3 Integrated Services (IS) with RSVP

The integrated services (IS) architecture [RFC1633, RFC2210] describes an architecture for providing the necessary network traffic control in order to enable mixed services (including real-time) to be supported in the Internet. The architecture is based on the assumption that bandwidth must be explicitly managed. This implies the need for resource reservation techniques and admission control. It is felt that simple prioritization is insufficient since, the network could become overloaded by too many high priority (real-time) users resulting in a poor service to all. A better solution is to offer a good service to as many users as possible, and restrict/prevent overloading by refusing additional service requests. IS achieves this by allowing users to request a desired Quality of Service (rate, delay etc). The network then decides whether this can be granted. The IS architecture comprises a number of components:

- **Packet scheduler.** This part is responsible for managing the forwarding of different packet streams using queues and or timers.
- **Classifier.** The packet scheduler makes use of a classifier to determine the class associated with each packet. Typically each stream/or class of packets is determined from the combination of source IP address and port number and destination address and port number. However systems could also make use of the TOS field, IPv6 flow labels etc.
- **Admission control.** This refers to the decision algorithm which determines if a new requested QoS can be granted or not.
- **Reservation setup protocol.** This refers to the protocols required to make a reservation request. The Resource Reservation Protocol (RSVP) [RFC2205] can be used for this purpose.

Currently there are 2 defined types of network service:

- **Controlled-load** [RFC2211] which provides a client data flow with a quality of service which closely approximates the QoS the same flow would receive in an unloaded network. This is achieved by using capacity (admission) control to prevent network overloading.
- **Guaranteed quality of service** [RFC2212] provides a service which guarantees both bandwidth and delays (within some bounds).

A potential problem with the IS architecture for scalable video applications is the explicit assumption that all the data for any one flow requires the same QoS. This implies that in order to support layered scalability, each layer will need to be contained in a different flow.

4.4.2.4 IPv6 flow labels

The new IPv6 protocol [RFC2460] includes a redesigned header which includes an 8 bit traffic class field and a 20 bit flow label which is intended for labeling sequences of packets for which a special handling is requested. The nature of the handling requested, is conveyed by an alternative protocol such as RSVP [RFC2205]. As yet it is not clear how the flow labels will or should be used.

4.4.2.5 Differentiated services (DS)

The differentiated services (DS) architecture [RFC2475] provides an new method for interpreting the 8 bit TOS field from IPv4 or traffic class field from IPv6, which under DS is referred to as the DS field. The first 6 bits of the DS field are used to describe a DS code-point (DSCP), while the remaining 2 bits are unused. The aim is that differentiated services would be used within a contiguous set of nodes with common service provisioning policies known as a DS domain. Within a DS domain the nodes make decisions on the per-hop-behavior (PHB) based solely on the DSCP. At the edges of a DS domain, more complex packet processing may exist including a combination of packet classifiers, droppers, shapers, metering, remarking and policing. These edge systems perform traffic conditioning to control the access to the DS domain, and may classify packets according to multiple fields including the DS field, source address, destination address, protocol ID and port numbers etc.

At present a few core PHBs are defined to correspond to certain code-points. These are broadly similar to those specified by the TOS field [RFC1349]. Other PHBs may be defined by different DS-domains.

4.4.3 Traffic Control for Scalable Video

Although there is a substantial amount of work based on transmission of real-time streams over an IP based network, much of this has been aimed at single layer non-scalable streams. In these circumstances the packets for the stream can all be assigned the same Quality-of-Service. Here RTP and integrated services architecture can be effectively used. The application first estimates the required bandwidth, delay constraints etc, and then makes a reservation request for these services. Alternatively, RSVP can be used to first determine the available quality of service choices and select an appropriate service, and configure the video application appropriately.

For scalable video the approach is much less clear.

4.4.4 2 Layer scalable system

Two potential strategies can be envisaged. Firstly the application could reserve an appropriate service for each layer (using RSVP and IS). Alternatively, the application could reserve a service for the base layer and transmit the enhancement layer using standard best effort transmission.

Alternatively differentiated services (DS) could be used by ensuring that the packets belonging to the 2 layers are allocated appropriate per-hop-behaviors (PHBs) within the network.

4.4.5 Multi-layer scalable system

For layered systems with more than 2 layers the situation becomes more complex. One strategy is to use integrated services and make an appropriate reservation for each layer. Alternatively, a differentiated services architecture could be used with appropriate per-hop-behaviors (PHBs) for each layer.

4.4.6 Fine-grained scalable system

For a scalable video application employing fine-grained scalability, the situation is far less clear. In order to make use of the fine-grained scalability, the system needs to support fine scaling of the FGS layer. This might be achieved by simple packet truncation. In order to enable this would require the implementation of an appropriate forwarding mechanism or per-hop-behavior (PHB). This might be done using a differentiated services architecture. A more realistic alternative is to perform the scaling within an application layer agent.

4.5 Proposed System for Scout

The main objective of the SCOUT video demonstration is the demonstrate the scalable video codec technology of A4.3.4 and A4.3.5 over a realistic reconfigurable radio system provided by the hardware validator of A4.3.3.

In order to reduce unnecessary effort on the part of both UoB (A4.3.4,A4.3.5) and FTR&D (A4.3.3), it desirable to keep the interaction between the different parts as simple as possible, without comprising the scope and realism of the demonstration.

In scout, we are primarily concerned with the reconfiguration and scaling for use over reconfigurable wireless systems. Thus although a real system is likely to comprise a combination of wired and wireless parts, we intend to concentrate mainly on the wireless portion. Thus the system shown in Figure 4-10 and Figure 4-11 can be simplified to that shown in Figure 4-12, while remembering that a complete system

should also support both the more complete wired/wireless situation and multicast applications with multiple receivers.

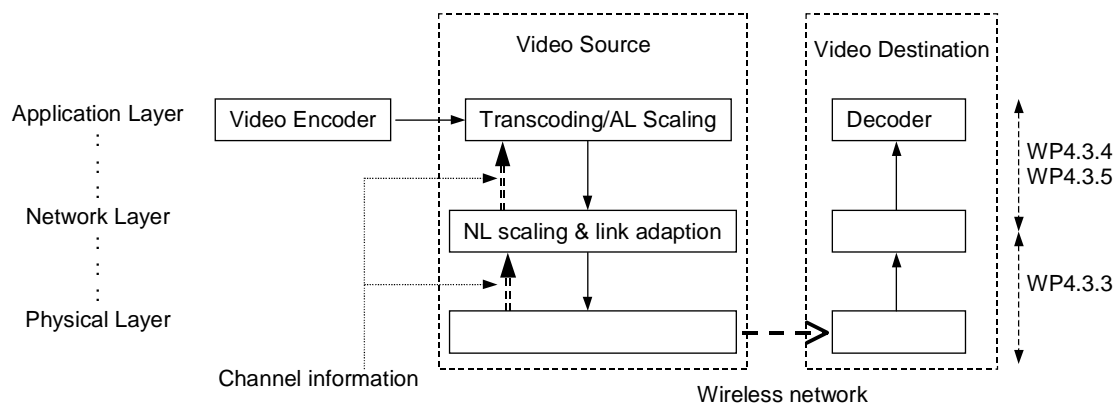


Figure 4-12: System to be used in Scout demonstrator.

4.5.1 Scalable Video Application Scenario for SCOUT Video Demonstration

In order to demonstrate the technologies developed within A4.3.3, A4.3.4, A4.3.5, the SCOUT video demonstration will focus on a non-real time unicast application such as might be employed in a video on demand system. This will provide a representative technology demonstration without the need for multiple channels and decoders, or tight constraints on time delay. The results achieved could then be easily extrapolated to multi-cast and real time applications.

However, it should be noted that although time delay will not be a critical factor in the SCOUT demonstration, it might be for some real-time multi-cast applications. Thus due consideration will be taken of unavoidable delays such as would be required by any retransmission protocol.

Figure 4.8 shows a block diagram for a typical non-real time uni-cast system, as will be demonstrated in SCOUT.

The encoder operates off-line to provide an archive of scalable video bit-streams which can be later transmitted. The scaling process is responsible for scaling/adapting in real time the bit-stream to suit the current transmission conditions. The network then delivers data to the decoder which also operates in real time. An optional feedback path would allow the end user (at the decoder) control over the scaling process and the material being delivered.

4.5.2 Packetized Video

It is envisaged that in a real world implementation, the video system (application layer) might interface to the physical layer communication system via an appropriate protocol stack (e.g. IP) comprising various layers including (from top down) RTP, UDP, IP, etc. Coupled with RTP/UDP/IP would be an appropriate QoS/traffic control techniques such as differentiated services and/or integrated services. Various extension protocols could also be added supporting non-standard features such as the delivery of corrupted packets.

For the SCOUT demonstration this would require the implementation/use of all the different protocols involved. This is a large amount of work which neither partner (UoB and FT R&D) are fully equipped or funded to undertake. Furthermore, before developing any non-standard extension protocols, the potential advantages need to be clearly demonstrated. This is one of the anticipated outcomes from the SCOUT video demonstration system. As a result of these factors, the SCOUT video demonstration system will use a much simpler direct link between the video application and hardware validator discussed below.

However, in view of the likely use of a packet based system such as IP within a final system, the intention is that the SCOUT video demonstration will provide use a packetized video stream. Each packet will also include a priority identifier, which can be used to indicate which video layer the packet belongs to. This priority identifier can then be used to determine which packets to transmit discard or truncate and to determine appropriate link adaptation and/or network parameters.

Note that video-packets are not inherently IP/UDP/RTP packets, but will be designed to be easily integrated into an IP/UDP/RTP system.

4.5.3 Simplified SCOUT demonstration system

Instead of implementing the complete protocol stack, it has been decided to build a direct interface between the scalable video codec software and the hardware validator. In order to minimize the effort required for the integration phase, the two parts of the system (video and validator) will be kept as separate as possible and communicate via a simple data pipe. This can be conveniently realized using TCP. Although data could be transmitted as a simple FIFO bit-stream, it is desirable to simulate the anticipated performance of a packetized system thus the video data will be split into a stream of packets.

The proposed approach has several key advantages:

- Simple to implement implying a high confidence of success, and requiring less additional effort from either UoB or FT R&D.
- More flexible, allowing the demonstration of aspects that current protocols may not allow (e.g. delivery of partially corrupted packets).
- Applicable to different protocol stacks, rather than being constrained to just TCP/UDP/IP.

The structure of the SCOUT video demonstration system is described more fully in the following section.

4.6 Proposed demonstration system

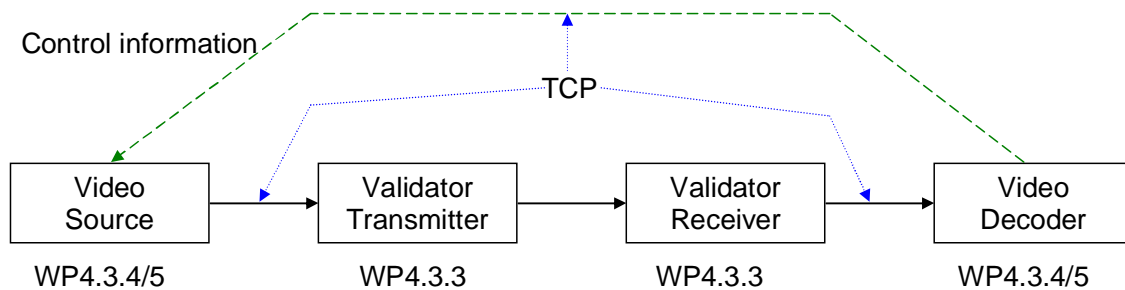


Figure 4-13: Block diagram of SCOUT video demonstration system.

The demonstration system is split into 4 main portions as shown in Figure 4-13 and discussed more fully in following sections.

- Video source.
- Validator transmitter.
- Validator receiver.
- Video decoder.

The control signals exchanged between these four parts are detailed in Figure 4-14.

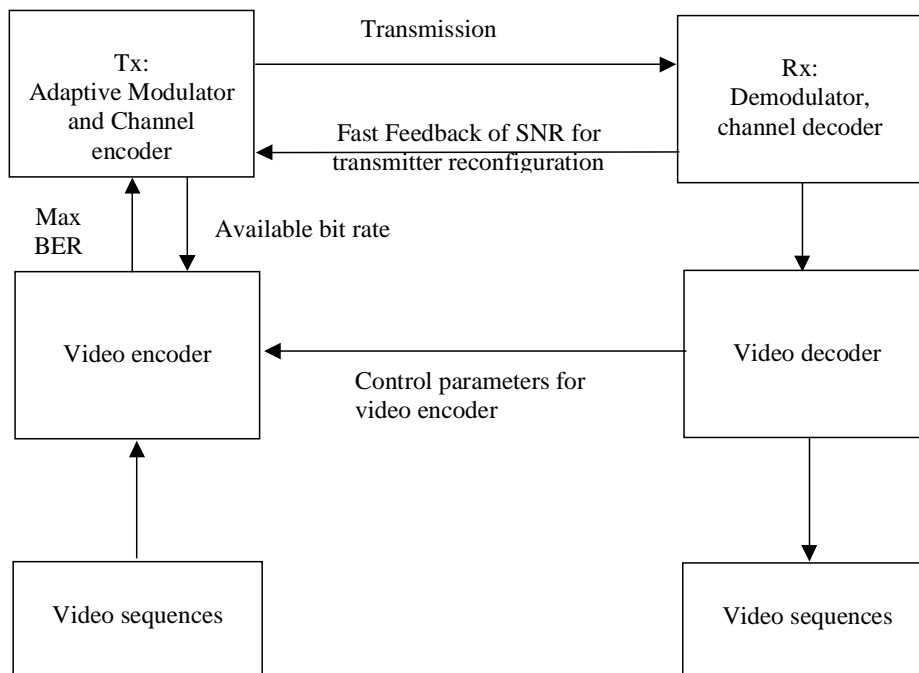


Figure 4-14: Interworking between hardware validator and video system

4.6.1 Connection different parts via TCP

It is proposed to keep these parts as independent as possible and to have a simple interface for passing data (in the form of video packets) between them. The proposed approach is to use TCP as bit-pipe for passing data between the component parts. This allows the code in the validator to be compiled and developed independently of the video. It also allows different video (or other) applications to be easily interfaced to the same validator, and different channel emulators to be easily connected to the video, without needing to recompile or make large changes to any code. Note TCP is used merely as a simple reliable bit-pipe for passing data between independent applications running on the same or different machines. In a real life system, both the video source/transmitter and receiver/decoder would be fully integrated as single applications. Thus the TCP connections are for experimental convenience only.

Although TCP provides a bi-directional connection, both the links between video source/validator transmitter and between validator receiver/video decoder, will be use as just uni-directional data flow.

One draw-back of using TCP to connect the different system components is an implied delay due to the buffering involved. For a non-real time application proposed, this delay is not a significant issue.

Note an optional additional feedback condition is included between video decoder and video source to allow control of the video source from the end decoder.

4.6.2 Location of video scaling process

One tricky issue to decide is where to place the scaling process. There are two alternatives:

- **Within the validator transmitter.** This allows the most direct adaptation to variations in capacity determined by the validator transmitter. However the use of any more complex scaling or rate control mechanisms would require significant code to be added to the validator.
- **Within the video source.** This approach allows greater flexibility in the encoding/scaling process. However it requires the knowledge of the current available capacity. This could either be signaled explicitly or determined implicitly based on the rate of consumption of data. The latter allows for a simple interface and would be easily applicable in a real life system.

Note the real issues here are which side of the TCP connection the scaling process should reside, and what control information the scaling process can make use of.

It is proposed to use scaling within the video source with data rate determined implicitly from the rate of consumption of video data. This can be done by using a buffer approach. The only difficulty here could be in possible delays and response times, resulting from the TCP connection and associated buffering between the video source and validator transmitter. It is unclear at present if this will prove to be a significant issue. If this proves to be a serious issue, then we could either:

- Ensure that video source and validator transmitter reside on the same machine thereby eliminating any delays due to wired network between the 2 parts.
- Implement a simple handshaking system over the TCP connection. I.e. the validator could acknowledge the receipt of each packet.

4.6.3 Information exchanged between system components

In order too keep the demonstration as simple as possible and thereby reduce un-necessary effort, the amount of information exchanged is to be kept as small as possible. The principal data exchanged is the video data it-self. This will comprise a sequence of video packets described in section 3. Any important information used by the hardware validator and video decoder will be included in a short packet header. This will include, packet length, time-stamp, packet type, QoS requirements, CRC etc. The main information of use to the validator is the packet length and QoS requirements. It has been decided that the most appropriate form for the QoS requirements (used by the hardware validator) is a desired maximum bit error rate (maxBER). Initially it was anticipated that the QoS requirements could be determined from the packet type by means of a look-up table. While this is a flexible approach which would allow extension to complex QoS requirements (such as max packet loss rate, or max packet delay etc.), it introduces extra unnecessary complexity to the SCOUT demonstration. Instead it is proposed to include a short maxBER field directly into the packet header, which can be easily read by the hardware validator.

Initially, it was thought that the video scaling would need some direct explicit feedback of the estimated available bit-rate from the validator. However, after further consideration it was decided that the available bit-rate could be estimated implicitly at the video source from the measured rate of consumption of video data. This is actually a less accurate but more flexible approach as it allows the use of other transmission systems (e.g. IP) which may not give any explicit information about estimated bit-rates. As a result no information needs to be explicitly passed back from validator transmitter to video source.

Similarly at the validator transmitter, no addition information (beyond the video packets) needs to be transmitted to the video decoder, as it is the decoder which will perform any CRC checking. However it may be of interest for the validator to display any relevant information directly.

4.6.4 Video Source

The video source will generate video packets at a rate to match that at which they are consumed by the validator transmitter. Each packet will have a fixed packet header described in section 3. This header will be designed to be easily read by the validator code.

The video source should determine the bit-rate at which the video is consumed by the validator transmitter. This can be achieved by time averaging to achieve a recent estimate of data rate. Having done this the video source will scale the video bit-stream to match this bit-rate estimate. There is some flexibility in the length of time to average over. A short time will lead to potentially larger quicker bit-rate variations. A longer time will lead to smoother variations. However a longer averaging time will also necessitate a larger transmit buffer and therefore larger delay to the video. It should be noted that this is an unavoidable delay. Thus for real time applications, a small buffer, and short time averaging is essential. The effects of varying the averaging time will be examined and demonstrated over the hardware validator.

The video source may also provide some additional loading to simulate other data such as other video streams, audio, web-browsing, email, data, etc. This will place more stringent restrictions (in terms of available bit-rate variations) on the video application.

4.6.5 Validator transmitter

The validator transmitter will receive packets of data from the video source and attempt to transmit them to the validator receiver.

For each video packet, the validator transmitter should:

- Determine the packet type and look up the corresponding QoS parameters: max BER, in a look up table. This will be used to control the hardware reconfiguration.

- Split packet into frames and/or slots as appropriate.
- Determine most appropriate modulation and error correction to be used for each frame or slot.

The validator transmitter should be able to handle empty buffer (TCP connection) on the input. This might occur if the available transmission rate is higher than the current data rate (from the tvideo source). The validator might handle this by inserting extra blank unused frames as appropriate.

4.6.6 Validator receiver

The validator receiver should receive and reassemble video packets. In the ideal case of an error free channel, these packets should match the packets at the validator transmitter. The validator should deliver packets in the correct order with no repetitions. The validator is not required to check the CRC or the integrity of the transmitted packets, and may deliver corrupted packets. The validator may discard any packets, but should if possible deliver them even if they are corrupted. This allows the video decoder to make a decision on whether to decode some or all of the data from corrupted packets.

4.6.7 Video decoder

The video decoder will decode and display the video data. The video decoder may also compare the decoded video data in order to calculate objective quality measures such as peak signal to noise ratio (PSNR). For this reason, each packet will contain a reference frame number to allow the decoder to find the corresponding frame in the original sequence. In order to allow the decoder to display the data at the correct rate, each packet will include a time-stamp.

Since the packets delivered by the validator receiver, may be corrupted (the CRC is not checked). The decoder should also examine the CRC for each packet in order to determine if the packet is corrupted. The decoder can then either:

- Discard the corrupted packet. This is the standard operation for an IP network.
- Attempt to decode the corrupted packet. For robust codecs, this may prove to provide improved performance, as the decoder can make use of portions of the packet which are not corrupted.

To allow the decoder to detect any missing packets, each packet will also include a sequence number which increases by 1 for each packet sent.

The decoder should also be able to display statistics relating to the number of packets lost and/or corrupted.

4.6.8 Graphical user interface (GUI) to video system

In addition to any validator control interface, there will also be a simple GUI to control the video system. This should enable control of the video source for adjusting both the video source and the scaling process. This may be placed in one of two locations.

- At the video source. This is the simplest approach.
- At the video decoder. This is potentially a more user friendly approach, especially if the two machines are physically separated. However, this approach requires an additional control link from the decoder to the video source, which can be conveniently, achieved using TCP sockets. Note that this return channel does not need to pass back through the hardware validator. This should be viewed as an enhancement to the demonstration and is not critical to its success.

At the current time, it is not clear what approach will be used as the initial effort will be focused on the more critical portions. However, the possible use of a TCP connection from video decoder to video source has been built into the proposed system (Figure 4-13) to allow for these extensions.

4.7 Software Interface

The proposed software interface will comprise a simple bit pipe between two independently compiled applications (video source and validator transmitter or validator receiver and video destination).

TCP can be conveniently used for this purpose. Note that the use of TCP here is purely a convenience and doesn't imply any implementation of the IP protocol stack, just the use of standard libraries for inter-process communication.

Data between applications will be in a packetized format, comprising packet header followed by payload data. The length of payload is determined from the packet length field within the packet header.

4.7.1 Video packet structure

Each video packet will comprise an integer number of bytes. The first 16 bytes (128 bits) will form the packet header (described below) the remaining ($L-16$) bytes will then comprise the data portion. The packet header will comprise the following fields:

- Bits 1-16. Total packet length L in bytes (used by validator and video decoder).
- Bits 17-32. Sequence number (used by video decoder). This will be incremented by 1 for each packet transmitted by the video source. It allows the video decoder to detect any missing packets. It could also allow the decoder to reorder packets if packets are received out of order. However, this will not happen with the SCOUT validator. Thus packet reordering will not be required. (This has the same roll as the sequence number in RTP header).
- Bits 33-64. Time stamp (used by video decoder). (This has the same roll as the sequence number in RTP header). For the SCOUT demonstration it is intended to measure time in ms. No reference time is required since it is the time-difference between packets time which is used. However we could use the standard PC time reference of midnight, January 1, 1970 UTC. Note that the time reference is the presentation time. Thus multiple packets may be given the same time-stamp if they belong to the same frame of data. (This has the same roll as the sequence number in RTP header).
- Bits 65-72. Packet type (used by video decoder). This is used by the video decoder to determine the type of data contained by the packet. It will also allow the interleaving of other data packets in order to simulate additional loading.
- Bits 73-80. $-10 \cdot \log_{10}(\text{maxBER})$ (used by validator). The requested maximum bit error rate can be calculated as $10^{-x/10}$. Thus a value of 10 corresponds to a maxBER of 0.1, and 20 corresponds to 0.01 etc. The validator may use this information to determine appropriate reconfiguration. This validator is not required to guarantee this performance.
- Bits 81-96. To be defined (used by video decoder) This field will include a frame number to allow the decoder to locate the reference frame for comparison purposes and calculation of objective quality measures such as PSNR.
- Bits 97-128. CRC (used by video decoder). This allows the video decoder to determine if the packet has been corrupted.

Note that all fields are stored in a portable (platform independent) big-endian format with high byte most significant bit first. Note that this is not the natural ordering for intel systems which use little-endian or (least significant byte first).

Since packets are transmitted between components by a reliable transfer protocol (TCP) no resynchronization is necessary. Thus the first byte transmitted will be the first byte of the header of the 1st packet. The first byte after the end of each packet is then the first byte of the next packet.

Packets can be read by first reading the packet header (16 bytes), then determining the packet length L and then reading the remaining ($L-16$ bytes) of data. This then leaves the process in right position to start reading the next packet. To exemplify this, example software will be provided (by UoB) in appendix A.

4.7.2 Control data passed back from video decoder to video source

As discussed in section 2, any control data passed back from the video decoder to video source is treated as an optional enhancement to the demonstration system and is not critical to success. Since it also does not involve the hardware validator the format used can be decided at a later date, and is not specified here.

4.8 Objectives of the Video demonstration

The video demonstration work is split into two main parts associated with A4.3.4 and A4.3.5 respectively.

4.8.1 Phase 1

A4.3.4 is primarily concerned with current standards based video coding techniques. In this context, we would wish to demonstrate the following types of operation:

- Single layer video with single QoS

- 2 layer (various types) video with 2 layers of QoS and packet by packet bit-stream scaling.
- 2 layer FGS video with 2 layers of QoS plus scaling within packets (by truncating packets).
- Systems with more than 2 layers of video and differing QoS.

4.8.2 Phase 2

A4.3.5 is primarily concerned with the development of improved systems. In this context we would wish to demonstrate (in addition to the aspects covered in phase 1):

- Adaptive variation of scaling operation to match users preferences.
- Delivery of partially corrupted packets for use by a resilient codec.
- Real time adaptation of encoding parameters to match user preferences.

4.9 Conclusion

This section provides a specification for the SCOUT video demonstration over the hardware validator. This requires integration of the video codecs (UoB A4.3.4 and 4.3.5) and the hardware validator (FT R&D A4.3.3). The aim is to provide a simple demonstration system which can effectively demonstrate the technology considered in A4.3.3, A4.3.4 and A4.3.5 without the need for a complicated interface or implementation of a complete protocol stack.

4.10 References

[Halsall] Data communications, computer networks and open systems. 4th Edition. F. Halsall, Addison-Wesley 1995.

[H.263+] ITU-T Recommendation H.263 version 2 January 1998 "Video Coding for Low Bit Rate Communication"

[IETF] The Internet Engineering Task Force home-page. <http://www.ietf.org/>

[MPEG-2] ISO/IEC 13818-2 Generic coding of moving pictures and associated audio, 1994.

[MPEG-4] Overview of the MPEG-4 Standard. <http://mpeg.telecomitalia.com/standards/mpeg-4/mpeg-4.htm>

[RFC791] Internet Protocol. J. Postel. Sep-01-1981. IETF RFC 791.

[RFC1633] Integrated Services in the Internet Architecture: an Overview. R. Braden, D. Clark, S. Shenker. June 1994. IETF RFC 1633.

[RFC1889] RTP: A Transport Protocol for Real-Time Applications. Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. January 1996. IETF RFC 1889.

[RFC2190] RTP Payload Format for H.263 Video Streams. C. Zhu. September 1997. IETF RFC 2190.

[RFC2205] Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification. R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin. September 1997. IETF RFC 2205.

[RFC2210] The Use of RSVP with IETF Integrated Services. J. Wroclawski. September 1997. IETF RFC 2210.

[RFC2211] Specification of the Controlled-Load Network Element Service. J. Wroclawski. September 1997. IETF RFC 2211.

[RFC2212] Specification of Guaranteed Quality of Service. S. Shenker, C. Partridge, R. Guerin. September 1997. IETF RFC 2212.

[RFC2250] RTP Payload Format for MPEG1/MPEG2 Video. D. Hoffman, G. Fernando, V. Goyal, M. Civanlar. January 1998. IETF RFC 2250.

[RFC2326] Real Time Streaming Protocol (RTSP). H. Schulzrinne, A. Rao, R. Lanphier. April 1998. IETF RFC 2326.

[RFC2429] RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H.263+). C. Bormann, L. Cline, G. Deisher, T. Gardos, C. Maciocco, D. Newell, J. Ott, G. Sullivan, S. Wenger, C. Zhu. October 1998. IETF RFC 2429.

[RFC2430] A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE). T. Li, Y. Rekhter. October 1998. IETF RFC 2430.

[RFC2460] Internet Protocol, Version 6 (IPv6) Specification. S. Deering, R. Hinden. December 1998. IETF RFC 2460.

[RFC2474] Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. K. Nichols, S. Blake, F. Baker, D. Black. December 1998. IETF RFC 2474.

[RFC3015] Megaco Protocol Version 1.0. F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen, J. Segers. November 2000. IETF RFC 3015.

[RFC3016] RTP Payload Format for MPEG-4 Audio/Visual Streams. Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, H. Kimata. November 2000. IETF RFC 3016.

[RFC3086] Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification. K. Nichols, B. Carpenter. April 2001. IETF RFC 3086.

[H.264] Editor's Proposed Draft Text Modifications for Joint Video Specification (ITU-T Rec. H.264 ISO/IEC 14496-10 AVC), Draft 7. *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG*

5. Middleware Demonstrator

5.1 Introduction

In the previous research project “TRUST”, the frameworks and systems needed to support software reconfigurable radios have been studied. SDR will allow mobile terminals to be reconfigured through software, enabling mobile devices to utilize different radio access technologies and communication protocols. It has been shown that the reconfiguration of SDR terminals requires complex interactions between the mobile terminal and the network entities, and occasionally even the download of new software modules to be installed on the terminals.

Following the Mode Negotiation & Switching Methodology developed in TRUST [5-1], an evaluation prototype for middleware technology is specified in this section that should provide demonstration and performance evaluation capabilities for the following main objectives:

- Middleware will locate a service, and establish a session
- Middleware will provide QoS Negotiation and Management
- Implementation of communication profiles of all relevant participants in a mobile scenario
- Middleware will show the Lookup/Trading Procedure
- Evaluation of Efficiency.

The middleware reconfiguration should support the following tasks and requirements:

- Establish connection with variable bandwidth (11, 5.5, 1, 0.1, 001 Mbit/s in WLAN) and different delays in server (5, 2, 1, 0.1 sec, no delay), which implies that delay/bandwidth vector is describing the dynamic network profile for a certain period of time during a running session.
- Combination of different bandwidth and delays for a connection managed by Admission Control & Resource Allocation in the server.
- Emulate the mobile channel according to the variation of bandwidth, delay vector during a session to, i.e. for each time period during a session we can assign a different bandwidth/delay vector.
- Definition of different applications types (downloads of HTML files, video and audio streaming). The streaming use cases will be emulated.
- Definition of suitable measurable QoS parameters of the running application, e.g. time to completely download all or a sub-set of objects of an HTML page, streaming bandwidth.
- Monitor QoS parameters during a session and post-process the measurements (delay, available bandwidth, average value of download time, variance and pdf of transfer and parsing time).
- Transfer the recorded and post-processed QoS to network for delay and bandwidth negotiations in the trading service.
- Mapping of user profile/ preferences and application profile in terms of delay and bandwidth to the present Network QoS Parameters (e.g. when low bandwidth I do not need pictures in HTML file, but only a short text description of suppressed pictures).
- Service Discovery and brokering in conjunction with the trading service.
- Bind to and use services.
- Location dependent & Context aware rendering of Service.
- Negotiate Quality of Service parameters.
- Map desired application QoS to Network Parameters – bandwidth and delay (i.e. independently implementing the QoS demands in end-user profile and applications).

The Demonstrator is consisting of the following functional groups/components:

- **Applications:**

A web browsing application will be implemented to demonstrate the most important concepts of the middleware technology.

- **Terminal Management:**

The demonstrator should support different types of end devices hence a terminal manager is needed that can handle different cases.

- **Simulation:**

A tool, which simulates different QoS, will be developed (adjustable bandwidth & delay).

- **Trading:**

Different service should be offered and selected by a trading service depending on complex preferences.

- **Reconfiguration:**

Services should be reconfigurable such that adaptation to changes in environment will meet user expectations and system requirements.

- **Evaluation:**

One of the aims of the demonstrator is to analyze the behavior of the system during the reconfiguration process. To evaluate the process, values of the effective delay (static delay & delay given by bandwidth) will be measured.

- **Profiles:**

To realise the trading manager as well as the reconfiguration controller e communication profiles will be used to describe all relevant actors (user, network, services, terminal) of a mobile scenario and hence the context (state of system environment) the system is in. This context information is used by the trading and reconfiguration processes as input for decision making.

The demonstrator will be implemented using the Microsoft .Net platform since it offers the necessary middleware functionality (web services, remoting, XML) and is significantly faster and easier to apply than the alternative J2EE platform.

5.1.1 Requirements from SCOUT D3.1.1

The former SCOUT deliverable D3.1.1 (Requirements for software methodologies for reconfigurability such as agents, middleware and adaptive protocols) has guided the requirement and design phase of this demonstrator and we repeat here briefly the most important requirements on middleware defined in D3.1.1.

Title	Time Predictable Middleware
Author	Martin Sénéclauze, CSEM
Version number	1.0
Description	The Middleware should enable real time applications to run this means that the duration of its interaction with lower layers should be predictable.
Technologies	Middleware
Priority	Required
SCOUT objectives	Facilitating hand over between modes

Title	QoS Support Middleware
Author	Bertrand Souville, DoCoMo
Version number	1.0
Description	The Middleware should monitor QoS parameters such as bit error rate, end to end delay between client and server.
Technologies	Middleware
Priority	Required
SCOUT objectives	Trigger to re-configuration

Title	Lightweight
Author	Bertrand Souville, DoCoMo
Version number	1.0
Description	Abstract middleware functions will be observable within the signaling functions. For a highly distributed architecture supported by middleware, network load and network latency might be unacceptable. Thus the middleware protocols and datatype encodings used for interaction between distributed components

	should be lightweight.
Technologies	Middleware
Priority	Required
SCOUT objectives	Limitations of wireless world

Title	Structured storage of user profile
Author	Thorsten Schöler
Version number	1.0
Description	The device's middleware should provide a structured way to store user profile data
Technologies	RDBMS, XML, etc.
Priorities	Required
SCOUT objectives	User, manufacturers and device profiles

Title	Device capability description
Author	Thorsten Schöler
Version number	1.0
Description	The device capabilities must be described and stored inside the middleware to provide web services data for delivering content, which is optimised for the device.
Technologies	CC/PP
Priorities	Required
SCOUT objectives	Best possible user experience of web services

Title	Structured storage of rules for decision making process
Author	Thorsten Schöler
Version number	1.0
Description	Due to the high degree of autonomy in decision making of a SDR device, data for the decision making process must be stored in a structured way inside the device
Technologies	RDBMS, rule sets
Priorities	Required
SCOUT objectives	Best possible user experience through high autonomy of the device

5.1.2 System Model for Reconfiguration

The Middleware is based on a high-level system model that abstracts various underlying hardware and platform specific aspects of a highly complex and distributed system. From the perspective of a high level system model, it is clear that high level applications are logical separated from the physical infrastructure in such way that there will be no need to directly communicate with each other. This isolates the more stable Control and Management aspects from the very dynamic technology thrust and commercial service needs, thereby creating enduring software Architecture. All the control and management functions are integrated into a unified, logical software architecture that is supported by a single Distributed Processing Environment (DPE). This means that instead of being forced to reside on a particular system at particular

geographical locations, the control and management functions can be flexibly placed in the network. A number of clear separation points, called Reference Points, are defined. They provide for clear separation of the roles of each player in the industry and will allow each player to enter the marketplace and flexibly expand its business.

The used system model, Figure 5-1, is focused on identifying necessary reconfiguration, controlling and management instances in both terminal and network from structural and logical points of view. Conceptualization of the system regards on dividing the network architecture into subsystems based on the nature of traffic (i.e. packet and circuit switch domains), protocols (i.e. access and non-access stratum, control and data) and physical elements (i.e. access and core network structure) [5-2]. Accordingly, Figure 5-1, builds on those bases to present a perspective of (Re)-Configuration Management (CM) instances, essential protocols and structural elements. Our focus in this document lies in the Middleware layer. Middleware is an open, broadly applicable and platform-neutral technology that uses object-oriented techniques to hide most of the underlying complexity introduced by differing systems. The benefits of incorporating a glue-layer are straight forward, considering the vast amount of different environments and access schemes and the subsequent proprietary protocols. Provided that adequate APIs are available, middleware copes with migration of objects from one platform to a remote platform, conforming a whole set of interworking functions.

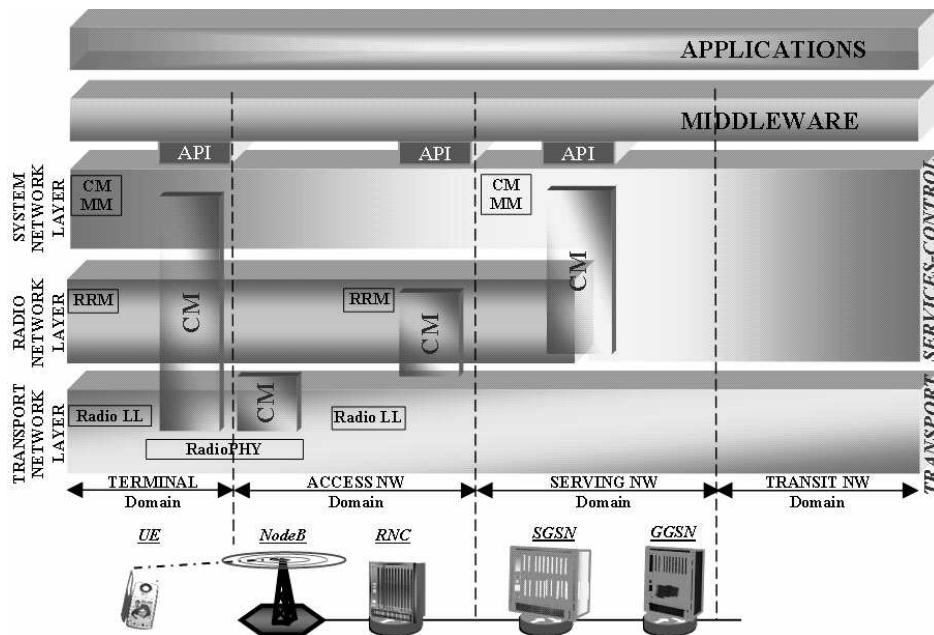


Figure 5-1: High-level Reconfiguration System Model

5.2 Requirements for the Middleware Demonstrator

Based on the underlying system model described in 5.1.2, middleware can be seen as an enabling layer of software that resides between the application and the networked layer of heterogeneous platforms and protocols. It decouples applications from any dependencies on the underlying layers, which consist of heterogeneous operating systems, hardware platforms and communication protocols.

Distributed object computing extends object-oriented programming by allowing objects to be distributed across a heterogeneous network, so that each of these distributed object components interoperate as a unified whole. These objects may be distributed on different computers throughout a network, living within their own address space outside of an application, and yet appear as though they were local to an application. Object-oriented middleware allows rapid development of platform-independent distributed applications. It is considered here for its potential in addressing the following issues related to software download for the reconfiguration of mobile devices:

- Locating of file server,
- Authentication of subscriber and software distributor,
- Capability exchange and transfer of software defined radio parameters.

5.2.1 General Objectives for the Demonstrator

Before requirements for the demonstrator can be analyzed, the objectives of the idea to “demonstrate” and “evaluate” a middleware concept need to be clarified. As stated, middleware is an enabling layer of

software that resides between the application and the network layer. This means that usually a middleware cannot be “demonstrated” the same as a new application, because ideal middleware is totally transparent to the user hence this not directly observable and therefore not directly measurable. Before thinking about how to demonstrate or evaluate hidden aspects and qualities of the middleware, it needs to be clarified what are the exact objectives of the demonstrator. The objectives of the demonstrator are classified into three groups:

- Visualizations: this group of objectives covers all objectives regarding visualization and demonstration of benefits of the whole middleware design, or single concepts that are part of it. Expected results are supposed to be made public without further interpretation, this means these objectives would speak for themselves (i.e. suitable for demonstration purposes in presentations, exhibitions etc.)
- Validation: the second group of objectives is conducted to gain practical experience regarding the used concepts and technologies. The results are not supposed to be made public in the first place but rather be used internally as feedback to the design process to improve the covered concepts, to test their functionality or to validate against a set of requirements or test cases.
- Evaluation: the last group are objectives concerning the evaluation of the whole middleware, i.e. gathering certain data that allows for measuring certain characteristics of the system that are crucial to make statements about the usability of the total system (i.e. the middleware and possible applications).

5.2.1.1 Evaluation of Middleware Concepts

Besides validating and verifying single key concepts as well as demonstrating functionality and benefits of all concepts described in later sections, statements about characteristics of the middleware concept i.e. statements about “how good” the developed concept (a middleware for reconfiguration) performs within its requirements specification cannot be made. Objectives regarding evaluation of the middleware concepts are differing from the proof of concepts and prototype aspects objectives (presented in section 5.2.3) because the demonstration results are not only used for internal purposes but will be published as part of the SCOUT project results, so they need to be meaningful also to people without detailed knowledge about the concepts, however technical knowledge about the domain can be assumed. The main objectives for this aspect are:

- Characterize the achieved results to practitioners in presentations
- Refine under-specified concepts by comparing different possible implementation and design decisions regarding their specification.

5.2.1.2 Measurements and Characterization of System Performance

This objective covers collection of ‘hard’ facts about timing and resource issues that can be used to allow statements about overall system usability and resource efforts necessary to introduce the developed concepts on a large scale. In particular answers to the following questions need to be found (each objective is numbered for later reference in the requirements analysis section, e.g. objective I1A etc.):

- Time to detect reconfiguration necessity
 - Detect a decrease in bandwidth below necessary level for a certain service (objective I1A²)
 - Detect an increase of overall response time (network + service load induced) above an acceptable level. (I1B)
- Time for reconfiguration (I1C)
- Time to choose between available services (modes) on an initial request in particular:
 - Time between receiving a request and finding a service (I1D)
 - Time between receiving the service reference and ready to use (binding time) (I1E)
- Resources needed to store one data element of a typical user, terminal, network or service profile (I1F)
- Scalability of all above measured values compared to number of concurrent users and services available. (I1G)

5.2.1.3 Refinement of under-specified concepts

Some concepts are under-specified because of lack of necessary data and experience, e.g. the distribution and amount of replication necessary to store profile data. This leaves several possibilities for design, implementation and deployment. Some under-specifications are necessary because of heterogeneity issues, e.g. some terminal might have sufficient resources to hold user profiles and take over part of the negotiation process, while others need to rely on resources of the network to accomplish this task. In both

² This is an identification of the described objective for later reference in the requirements analysis. [Main objective group I, II or III][Subgroup 1.x][Objective A, B, C...X]

cases it is necessary to measure certain parameters that make the alternatives comparable. For the following under-specifications in concepts different possible solutions need to be characterized:

- Distribution of profile data (I2A)
- Specification of user preferences (by element, rule based etc.) (I2B)

5.2.2 Visualization Objectives

Besides the main objectives of the middleware concept, evaluation of some secondary goals for the middleware demonstrator exists. As initially stated; middleware is usually invisible and therefore cannot be easily demonstrated as for example an application which can be presented by just using it or let interested persons play around with it. Middleware is usually neither explicitly “used” by a user of an application, nor the developer of an application. It is “used” by software artifacts to simplify complex communication between each other or easily access resources by automating reoccurring tasks. This allows software modules developer to concentrate on the development of functionality rather than organizational tasks and comparing the utilization of the existing infrastructure of functionality that is not related to a certain task for which a software module is developed and therefore common for most modules like for example finding resources in a mobile network.

For the middleware demonstrator four main objectives regarding the demonstration and presentation aspect of the demonstrator exist:

- The functionality of the overall middleware concept.
- The functionality of certain key concepts.
- Visualizing of certain interesting new/key concepts.
- Demonstrating the benefits of using Middleware technology in Mobile communication environment.

5.2.2.1 Demonstration of the overall middleware concept

Demonstration of the complete middleware concept would involve a visible verification of each defined requirement for the middleware by presenting successful runs for one or more test cases for each requirement. For a system with a certain complexity this will lengthen the time used for presentation purposes and gets boring. However from the initial requirements analysis, only a certain number of application scenarios will be presented/demonstrated Corresponding to the user requirements scenarios one application should be presented. The main objective here is to show that the scenario works as it was supposed to be and draw attention to the presentation. It is much more important for the application to be interesting rather than demonstration any key concepts. The objective here, is:

- Presenting one fancy application that is related to one of the scenarios in the user requirements analysis (II1A)

5.2.2.2 Demonstration of important key concepts of the middleware

Rather than demonstrating and verifying every single requirement, important, new or key concepts of the middleware should be demonstrated. This is especially difficult regarding to the invisibility of most of the middleware aspects. However most concepts can be demonstrated indirectly with suitable test scenarios, for example by artificially simulating certain conditions while running a special application and then watching certain side effects or their absence. A general example for the absence of side effects could be that cutting the connection to a server does not kill the application).

The objectives here are to demonstrate the following key concepts within the middleware:

- Finding and using services (II2A)
- Service Trading/Brokerage based on (each type of) communication profiles (includes QoS Negotiation) (II2B)
- Adapting an application to a sudden change in network QoS (II2C)
- SADL usage (II2E)
- Service Download (II2F)
- Management of Profiles (II2G)
- Handling of wireless link interruptions (II2H)

5.2.2.3 Visualizing concepts

Visualizing indirectly observable concepts work is hard and even harder when presenting or demonstrating how they work. While it might be possible to simulate a problem of a certain concept by using an adequate scenario, this will usually lead to misinterpret the target of a demonstration and how things are accomplished. Nevertheless it might be necessary to support presentations of certain concepts as part of the accomplished results of a given project, especially the target audience of a presentation lacks the necessary technical background to understand a concept in full detail.

The objectives here are to show, what is going on ‘under the surface’ of the following concepts:

- Finding and using services (II3C)
- Service Trading/Brokerage based on (each type of) communication profiles (includes QoS Negotiations) (II3B)
- Adapting an application to sudden changes in the network QoS (II3C)

Note that these objectives might generate requirements that are not only related to the technical specification of the demonstrator, but also the specification of the whole demonstration and presentation specification including speech scripts, handouts, flyers, presentation slides etc.

5.2.2.4 Demonstration of the overall middleware concept benefits

The most important part of every new concept presentation besides proofing that a concept is functioning, is the demonstration of the benefits of the overall concept or at least of each key concept considered. Beneficiaries in this case could be the customer/user, network operator and service provider. A benefit could be, that the system applying the concept can achieve something new or can do something 'better' than before. New in this context means enabling new functionality i.e. new applications that could not be implemented before, 'better' usually boils down to some existing functionality which can be realized cheaper or with higher quality, e.g. faster, than before.

The major benefits of the demonstrated concepts will cover both the 'new' and 'better' aspect, since this is one way of realizing certain desired new functionality that might have some side effects falling into the 'cheaper' category. However during the overall development time of the demonstrator there might merge new concepts or available solutions of single aspects of the desired functionality or smaller subsets thereof might be discovered. In these cases it needs to be demonstrated that the provided overall solutions is better or at least not worse than these special aspects. Therefore possible demonstrations of concepts contained in subsets of new functionality need to be constantly monitored for their availability and adequate comparison demonstration strategies need to be developed 'on the fly' that will show the superiority of the demonstrated concept, compared to the competitor within a subset of the 'new' benefits. The objectives here are to demonstrate the following benefits, grouped by their target:

- Customer benefits
 - Location Based Services (II4A)
 - Central managed user preferences (II4B)
 - Device independent usage (II4D]
- Network operator benefits
 - Load balancing (SADL) (II4C)
- Service provider benefits
 - Software Update (II4D)

Note that in some cases the identity of the network operator might also be the (main) service provider and might have objections against (full featured) third party service providers. Therefore the developed demonstration scenarios should contain an alternative branch for such a case that can be applied or not, depending on what the target audience objectives would be.

5.2.3 Proof of Concepts & Prototype Aspects

The overall idea of prototype demonstration is not only to publicly visualize concepts, but also to gather practical information for feedback purposes into the development process. Even if the theoretical developed concepts were verified against the requirements, an actual design or even implementation attempt might show that the concept is too difficult (in terms of resources necessary) to realize or can be improved. However in most cases the reason is that some concepts are under-specified and therefore cannot be completely verified especially regarding to nonfunctional requirement constraints (e.g. timing). Implementing a concept is one possibility to remove under-specification, i.e. deliver a complete specification.

The three main objectives regarding the demonstrator aspect of collecting data for internal feedback purposes are:

- Getting data about difficulty of a concept implementation
- Verify a concept
- Validate requirements connected to a concept

The three groups differ from each other in terms of to what extent a concept will be implemented.

5.2.3.1 Getting data about difficulty of a concept implementation

To estimate the difficulty of implementation and to get an idea of how much effort will be necessary to make a full-scale implementation, a partial implementation of the concept is sufficient, as long as it is possible to identify the new implementation tasks within a concept that cannot be covered by existing component libraries, infrastructure frameworks or well-established patterns. Usually there is no need for a

complete implementation that works for even only one possible scenario. Instead of that, an architectural design is usually sufficient.

If the design envisions the use of new technology, implementations of examples of this new technology's usage might be necessary but these examples need not be related to the concept, i.e. are only necessary to gather general experience with the new technology.

Concepts that should be inspected regarding their difficulty of implementation are:

- Service Brokering (III1A)
- Runtime Adaptation (III1B)

5.2.3.2 Verification of Concepts

There are several possibilities to verify a concept, which should prove that the concept meets the requirement specification. One possibility of verifications is to implement previously defined test cases and run them against their specification. Other possibilities involve formal methods and proofing methods. A requirement analysis in 5.2.5 will show which methods to be applied. However all methods share the fact that only the core functionality needs to be specified or implemented that in fact is a kind of specification. There is no need for helping functionality like user interfaces etc. The concepts to be verified are:

- Brokerage (III2A)
- Adaptation (III2B)

5.2.3.3 Validation of Concepts

Validating means checking whether a concept meets the defined requirements and also meets the demands of the real world, in other words finding out whether the requirements are wrong by testing the implementation of a concept against the real world, for example by letting a representative number of test users use the implementation. Of course the implementation first needs to be verified against the concept and the concept against its requirements. If this is not the case, then the validation also contains the verification, which means in case of a misbehavior it need to be found out case by case whether it is caused by wrong requirements, errors in the concept or errors in the implementation.

Therefore it is not enough only the core functionality to be implemented but also the helper functionality like:

- User Interface for accessing service based applications (Service Browser) (III3A)
- User profile management (III3B)

5.2.4 Summary of the Objectives

In the last section different objectives and motivations that describe the purposes demonstrating the Middleware concept has been discussed, summarizing there are three different kinds of motivations:

- Objectives that try to publicly visualize achieved results for both non-technical consumer and a technical audience.
- Project internal objectives to test certain key concepts for both optimization feedback and verification/validation purposes.
- Objectives of the third type cover measurement of concept and overall system application characteristics to discuss performance issues and evaluate competing concepts.

The following table cross-references each objective with their motivations.

Table 5-1: Objectives Summary

Concept	Visualize			Proof			Measure	
	Demonstration	Visualization	Benefits	Dev. Feedback	Verification	Validation	Characteristics	Refinement/Evaluation
a) Overall System	X							
b) Service Usage	X					X		
c) Service Finding	X	X					X	
d) Service Trading	X	X		X	X		X	

e) Service Adaptation	X	X		X	X		X	
f) Service Download	X		X					
g) Profile Management	X		X			X	X	X
h) SADDL	X		X					

5.2.5 Selection of Appropriate Demonstration Scenarios

The previous sections briefly describe all objectives that are involved in demonstrating a concept for reconfigurable middleware. For the middleware demonstrator there are four main objectives identified, regarding the demonstration and presentation aspect of the demonstrator:

- The functionality of the overall middleware concept.
- The functionality of certain key concepts.
- Visualizing of certain interesting new/key concepts.
- Demonstrating the benefits of using Middleware technology in Mobile communication environment

How good these objectives can be fulfilled highly depends on the kind of applications that are demonstrated since the middleware itself is mostly transparent for the user and therefore any potential observers.

In contrast thereof, success with other concept proofing or measuring objectives depends usually on certain coverage of the demo application. These objectives can be fulfilled by any application that makes use of all relevant parts of the middleware to be evaluated. Therefore significant effort needs to be put into selecting an appropriate application and demonstration scenario, of course taking into account the resources available for development of the demonstrator, which are very limited in this case.

The best way to fulfill all objectives is to present a completely new type of application that has a clear customer benefit and is based on the new middleware concept which covers all four motivations related to demonstration and visualization issues at once. An example for such a scenario can be found in [5-6]. However this application is in fact a meta-application that eases access to a convenient adaptive selection of sub services. This implies the availability of a larger number of application scenarios and is therefore out of scope of the resources available for development of this specific demonstrator.

A more reasonable approach is demonstrating two scenarios that complement each another. The first scenario demonstrates/shows a well-known type of application using the middleware concept such that no negative effects on existing applications will occur. Further, the first scenario will be spiced up with improvements of each key concept of the middleware concept. These improvements can be switched on and of to reflect the benefit of a middleware. Since each key concept is covered, the necessary coverage for the proofing- and evaluation objectives is also fulfilled.

The second scenario is more a showcase and should demonstrate one interesting application that might be well known in another domain than mobile communication (e.g. known from desktop based internet), but not possible to implement in a mobile communication environment so far. The main purpose of this scenario is to grab attention of the audience and therefore does not have to cover all key concepts or show a detailed benefit for all groups, as long as it makes use of some of the possibilities of the middleware concept to be demonstrated.

A visualization engine like described in [5-7], [5-8] and [5-9] and used in a similar demonstrator can support/complete the two scenarios for visualizing aspects of the middleware concept that are too difficult to be visualized /shown indirectly by using an application. Note that this is a purely optional scenario that will be implemented only if enough resources are available for development.

5.2.5.1 Surfing the Web

This scenario is a spiced up version of the well-known web browsing application type. The main purpose is to demonstrate that a common mainstream application can be carried out on the middleware infrastructure without any negative side effects for e.g. resource usage and scalability.

The user connects with a full-featured client (a notebook) to a proxy server that offers an Internet gateway service and browses any web page he likes. After some time a reduction of network bandwidth is emulated. The client notifies the network about this reduction, which let the current service adapt to the new situation by reducing quality of transferred pictures or simply strip the browsed web-pages of all multimedia content. Alternatively the network could hand over the browsed session to a second available internet gateway that is specialized in serving low bandwidth optimized web pages, e.g. by using compressed WML instead of HTML.

Afterward the user switches to another client and can continue browsing the web from the point he left on the notebook (but using a different screen size). He starts a download in a high bandwidth situation. A reduction in bandwidth to 10kbit/s simulates the user leaving the WLAN Hotspot to GSM. The user then

decides he needs his download to finish quickly. Therefore the client sends an active request to the network for reconfiguring to higher bandwidth and switching the session to UMTS (emulated by increasing network bandwidth to 384kbit/s).

On the optional visualization, the different gateway services that are represented as boxes in the client (see Figure 5-17 where the major components of the middleware are illustrated) can be monitored. Interested observers can monitor network bandwidth and delay bar and watch notifications sent into the network and the reconfiguration reaction in form of re-wiring communication channels between the service boxes.

With this scenario all main concepts (b. to h, see Table 5-1) can be demonstrated since it involves finding and using an appropriate service using the service-trading concept. The network reconfigures several times depending on the user profile, for instance if the user wants to have a quick download instead of slow and cheap one.

The SADL deployment is a bit more complicated to involve, but could be shown by simulating an overloaded gateway service that could lead to powering up and deploying a second service on a reserve server or more simple be load balancing a second client to another already running service.

5.2.5.2 Re-configurable Streaming Media Emulation

This scenario is a simulation of a generic streaming media application to represent the class of (broadband) streaming applications that have a usage pattern that is different from that of the interactive web-browsing scenario.

This scenario can represent a showcase and demonstrates an interesting application that is well known within the domain of desktop Internet usage but not yet available for wireless mobile clients in a heterogeneous environment. Because of resource restrictions for demonstrator implementation this scenario can only be emulated for evaluating a different usage pattern. A fully functional implementation is out of scope.

5.2.6 Analysis of Demonstrator Requirements

In this section a detailed analysis of the requirements is given that is based on the detailed description of the demonstrator objectives and a short overview about selecting appropriate scenarios to fulfill them. Further requirements can be

The requirement analysis is grouped into four areas; each of them represents one main aspect of the overall demonstration system:

- Demonstrator requirements
- Middleware requirements.
- Applications requirements.
- User interface requirements.

This follows the usual approach of modularizing business applications into a presentation, business logic and an infrastructure layer (i.e. separating both the user interface code from the application functions code and application specific code from code realizing middleware functions).

5.2.6.1 Requirements for Demonstration and Evaluation

This section describes requirements that are connected to the demonstration or evaluation issues only, i.e. they are neither motivated by the object that will be demonstrated (the application scenarios in our case) nor are they a technical necessity to enable the demonstrated object (the middleware). They are only based on the objective to demonstrate or measure certain parameters. All requirements related to visualizing hidden aspects and measuring system parameters for evaluation or testing purposes fall into this category.

5.2.6.1.1 Network Bandwidth Control

Description: Within the Demonstrator there are needs for possibilities to control the effective bandwidth that can be used by the various services in the demonstration system. Effective means that it is not necessary to change the bandwidth in the physical layer, but it can also be artificially reduced or increased within the hardware capabilities, as long as the throughput between services and service endpoints is controllable.

Reason: This is because concepts d, e, f and h are related to quality of service aspects, especially concept e, and network bandwidth is the most important and most easy to understand service and quality and therefore most suitable for demonstration purposes to an audience not acquainted with technical details (see especially 5.2.1.1), quality of service parameter. The previously mentioned concepts are involved in visualizing objectives (II1A, II2B, II2C, II2E, II2F, II3B, II3C, II4C, II4D) and can serve as an example in proofing concept objectives (III1A, III1B, III2A, III2B) as well as serving as reproducible and controlled measurement of performance values in the evaluation concept objectives (IIA, IID, IIE, IIG).

Verification: This requirement can be verified by a test case for each used transportation protocol that downloads data of a given size and cross checks download time to the set bandwidth.

Constraints: The control can be discrete as long as the steps are meaningful, i.e. 10kbit/s (GSM), 384kbit/s (UMTS) etc. The control range must be at least between 10kbit/s and 384kbit/s.

Reason: The demonstrated bandwidths should be meaningful for the addressed audience of demonstration and show typical used bandwidths within the project domain (wireless mobile communication).

5.2.6.1.2 Network Delay Control

Description: Within the Demonstrator there is a need for possibilities to control the effective delay of the communication between various services in the demonstration system. Effective means that it is not necessary to change the delay in the physical layer, but it can also be artificially reduced or increased within the hardware capabilities, as long as the time between sending and receiving a message between services and service endpoints can be controlled.

Reason: This is because concepts b, c, d, e and h are related to aspects of response time and the network delay is one of the two components that influence response time (besides the service message processing time). The network delay is the easiest way to observe the quality of a distributed application and therefore most important for evaluation purposes of the overall system practicability and usability (see especially 5.2.3 and 5.2.3.2 concept proof and evaluation objectives). Network delay is involved in visualizing objectives (II1A, II2A, II2E) and necessary for controlled measurement of performance values in the concept evaluation objectives (I1B, I1D, I1E, I1G).

Verification: This requirement can be verified by a test case for each used transportation protocol that is connected to a dedicated service (like ping) and measures the round trip delay of a reflected test-message. The results can be compared to the set network delay that shows that the artificial network delay control is working properly.

5.2.6.1.3 Network & middleware Visualization (Optional)

Description: Within the Demonstrator the hidden aspects that cannot directly or indirectly be observed should be visualized by a means or another.

Reason: This is because objectives II3A, II3B and II3C need some additional visualization that goes beyond showing application scenarios for clarifying in technical presentations how certain key concepts work. Moreover during the system development and for concept verification objectives III2A and III2B, this kind of graphical debugging capabilities comes in handy (see also [5-8]).

Verification: This requirement can be verified in comparing the graphical output to the corresponding concept specifications (Message Sequence Charts, Object Diagrams etc.).

5.2.6.1.4 Evaluation of Measurement Data

Description: The necessary data for statically evaluation can be gathered either on the fly “online” or can be stored for later processing. The online data can be evaluated during demonstration carried out in the lab whereas the pre-stored data are used for demonstration on conferences and another purposes. How to gather the data and which parameters to be included also the mechanism for varying the bandwidth (e.g. inserting dummy data) and the switching time between constant bandwidth and variable bandwidth will be discussed in detail in the specification phase.

Reason: The gathered data is needed for collecting ‘hard’ facts about timing and resource issues that can be used to allow statements about the overall system usability and resource efforts necessary to introduce the developed concepts on a large scale (see objectives I1A-I1G).

Verification: This requirement can be verified by availability of necessary measuring data. The accuracy of measured data need to be verified against test cases describing alternative time based measuring strategies for each parameter where possible (e.g. sending a test file of known length, and external measuring time calculating bit rate comparing to internal measured or known bit rate.)

5.2.6.2 Functional Middleware Requirements

To demonstrate a middleware concept by implementing typical applications which make use of the middleware will implies the necessity to implement also the middleware concept or at least the parts that the application scenarios make use of.

In this section the requirements connected to theses aspects of the middleware that are necessary to be implemented for the demonstrator are described.

5.2.6.2.1 Service Broking/Trading

Description: Application services will expect the middleware to act as an authority that they can consult which will deliver the partner details directly or list the possible candidates. It will act as a simple directory, delivering just the set of intermediates that might finally direct the client to the requested service with an implied negotiation between the parties who offer the services and using the network

centric quality of service functionality described in the following requirement ((Re-) configuration Decision below).

Polices and Strategies of trading:

What plays a role in trading is yet not clearly defined, but if from the users point of view in trading delay is the most important point or decision. Users cannot tolerate long delays and want to access/use the service in less time as possible. However delay is composite parameter (static & dynamic), it is possible that to alter the static delay for high priority user (e.g. divert the traffic to less loaded server). On the other hand, the application point of view in trading is based on the bandwidth. Therefore policies/strategies for trading should be a combination set of both views. The details of the trading strategies will be included in the design specifications documents

Reason: The reason is that having a dynamic system structure with reconfigurable terminals travelling through a changing environment of changing service availability moreover potentially restricted by dynamic requirements from many sides (billing preferences, network availability and load etc.) will lead to have serious impact/influence on the application as there need to be a way to find and choose services they depend on according to their changing availability (e.g. the terminal might leave a network and some services are not available any more). Services that can be found might be services not yet available during the application service's development time, having additional functionality that they can contribute to.

Verification: The scenarios described in 5.2.5 are test cases to verify proper functionality of this requirement. Since this is a demonstrator that shows scenarios, which are based on a middleware concept and not yet fully implemented, it is sufficient to proof that the implementation only works for the demonstration scenario. The implementation does not have to work for other possible scenarios.

5.2.6.2.2 (Re-) configuration Decision

Description: There is a need of an automatic decision unit that monitors all Quality of Service parameters, which are involved in an actual situation of a service-based system. By mapping the user profile/ preferences and the application profiles of delay and bandwidth terms into the present Network QoS Parameters, conclusions can be drawn about the necessary dynamic requirements (e.g. when low bandwidth I do not need pictures in HTML file, but only a short text description of suppressed pictures) and initiate their implementation by reconfiguration.

Reason: The reason is that not all possible requirements in applications lifetime can be implemented by simple superposition in a static system structure, especially if resource constraints within the system exist. For example it might be technically possible to construct terminals with three different built in radio interfaces, but 6 built in interfaces might not be reasonable any more regarding complexity.

Reconfiguration can help to solve this problem since it can reduce the systems complexity at each point in time but of course it also adds additional complexity over time (e.g. if one allows for more than one reconfiguration at the same time within the system).

Having more than a few possible reconfigurations or reconfiguration needs to take place within small time segments, the reconfiguration to meet a new dynamic requirement cannot be done manually or interactively by the originator (e.g. the terminal user).

Optional Constraint: Capability to negotiate based on the Composite Capability/preference profile (CC/PP).

Verification: The scenarios described in 5.2.5 are test cases to verify proper functionality of this requirement. Since this is a demonstrator that shows scenarios, which are based on a middleware concept and not yet fully implemented, it is sufficient to proof that the implementation only works for the demonstration scenario. The implementation does not have to work for other possible scenarios.

5.2.6.2.3 Adaptation Negotiation Protocol

Description: There is a need for a protocol that defines a negotiation between the reconfiguration decision unit and service using clients/services within the system to decide whether a certain change in the system environment (e.g. bandwidth, delay) can be handled/tolerated by the affected services and clients. This requirement is optional. The necessary specification can be found in the Appendix because it represents a second approach for reconfiguration that is orthogonal to the rest of the demonstrator requirements.

Reason: The reason is that reconfiguring the system might not always be the most efficient way of adapting to certain changes of environment parameters. Services might be able to change their behavior themselves to adapt to certain changes (like e.g. dropping pictures in low bandwidth situations) or clients might tolerate violation of QoS contracts for a limited time. It is necessary to look into this additional adaptation strategy (besides reconfiguring the system) to get evaluation results about the overhead that is introduced by the (much) more flexible concept of adaptation by system reconfiguration.

Verification: The scenarios described in 5.2.5 are test cases to verify proper functionality of this requirement. Since this is a demonstrator that shows scenarios, which are based on a middleware concept and not yet fully implemented, it is sufficient to proof that the implementation only works for the demonstration scenario. The implementation does not have to work for other possible scenarios.

5.2.6.2.4 Application-Level Service Adaptation

Description: In some cases the applications are forced to be disconnected or have to be recomposed such that certain requirements in a changed environment have to be met. In such situations applications are offered a chance to take corrective actions to address those changes themselves. Application-level service adaptation is an intermediate step in taking corrective action with respect to application's context changes such as communication bandwidth or delay. This requirement is optional. The necessary specification can be found in 5.5.6 for the same reasons as described in 5.2.6.2.3.

Reason: Assuming that middleware-initiated reconfiguration (breaking up an application and recomposing it) is more expensive than application-level adaptation, adaptation can serve as valuable complement to reconfiguration.

Verification: Time for reconfiguration needed for both adaptation and reconfiguration is to be compared.

Note: Transitive component-level dependencies and resulting conflict situations have to be addressed in production systems, but will not be taken into consideration in the demonstrator.

5.2.6.2.5 Communication Profiles

Description: To realize an optimal service negotiation process the different "actors" of a mobile system that are described by communication profiles should be taken into considerations. A communication profile is an abstraction of all relevant participants in a mobile scenario, as there are the device, its network environment, the user and the available services on different levels ranging from lower layer bearer services to higher level teleservices and application services, the profiles contain all relevant information that will be needed for choosing a certain service. For the middleware demonstration, it is important that the user and application profiles should be implemented.

User profile:

A delay/bandwidth (data rate) vector that describes the user profile/ must be implemented. Users can be classified according to criteria such as preferred applications behavior in case of QoS adaptations, billing preferences, etc. In addition the same user will have different interests and requirements to services depending on his personal situation. These user requirements/demand according to the different delay/data rate vector are the triggers in the trading services, one example can be that a certain user A will demand a certain range for data rate and consequence certain delay. For example let r_1 be the minimum rate that user A will allow, r_2 be the maximum rate he/she can afford according to the billing preferences, d_1 be the minimum delay the network can offered to user A and d_2 be the maximum delay the use A can tolerate. Therefore for user A:

$$r_1 \leq U_{A,R} \leq r_2 \text{ and } d_1 \leq U_{A,D} \leq d_2$$

Hence $\vec{V} = (r, d)$

Therefore *user profile* $\hat{=}$ $\{\vec{V}_1, \vec{V}_2, \dots, \vec{V}_n\}$

User Profile data:

- Preferred data rate (dynamic delay) and static delay range described by a vector, which results in an effective delay for the web browsing
Static delay models the CN behavior and dynamic delay models the currently available bandwidth in the RAN:
- Business user: low static delay and always highest bandwidth wanted
Best effort user: no negotiation possibilities, i.e. must accept every QoS

Application profile:

The application profile is determined by the different application behavior towards the different QoS offered by the network. The application will adapt according to the QoS, an example for that would when downloading a HTML file that contains pictures. In the case that the network had offered a high data rate, i.e. less delay, the application may decide to download the complete file. On the other hand if the offered bandwidth is not sufficient to download the whole file and the delay will increase accordingly the application may decided to adapt to this situation by either pre-process the picture (down sampling) or leave them completely out and download only the text. Different possibilities for the application behavior are:

- 1st possibility: Regular behavior that is independent from current QoS.
- 2nd possibility: Adaptive behavior at high (>384kbit/s), medium (64 – 384 kbit/s) and low bandwidth (10-64 kbit/s).

The Application behavior at different bandwidth can for instance be:

- At high BW: the application behave as in 1st possibility described
- At medium BW: the application will reduce picture size (e.g. sub-sampling them and yielding reduced resolution)
- At low bandwidth: the application will replace original pictures by small icons

- To summarize the user profile and the application profiles are the triggers/inputs to the trading service. The important point for decision from the user point of view is the delay experienced. Delay is composed of two parts the static delay and the dynamic one. With static delay we understand the delay imposed by the server, we can we artificially delay the web browsing to emulate the core network (CN) delays caused by routers and other network elements. Possible values of the static delay are: NO delay, 1, 2 and 3 seconds.

The dynamic delay is defined over the download traffic and currently used bandwidth: Therefore the following BWs are required: 10, 56, 144, 384, 1000, 11000 kbit/s.

The effective or composite delay is the sum of static and dynamic delay. The application on the client can only measure the composite delay, whereas the network part is able to differentiate the static and dynamic delay.

The following tables give an overview of the different user profile and application profile.

Table 5-2: User Profile

	Date Rate/Bandwidth	Delay	Priority
Business man profile	Demands the maximum rate can be offered	Less delay as possible (preferable only the network latency)	1
Medium class profile	Restricted to the maximum rate defined in the preferences (Billing reasons)	Can tolerate delay	2
Cheap class profile (student)	Can accept any rate offered by the network and his priority to negotiate is minimum	Can wait forever	3

Table 5-3: Application Profile

Cases	QoS offered	Action taken by the application
Case 1	High data rate, less delay	Successful download in reasonable time
Case 2	Medium data rate/tolerable delay	Adapting the application to the offered QoS
Case3	Low data rate/high delay	The application in this case is not taken any action it is up to the user to accept this service or to rejected.

In summary the communication profiles contain all relevant information that will be needed for choosing a certain service for an interaction or reconfiguration during a service interaction. Examples for profile information are capabilities of the current terminal, user preferences and current network bandwidth.

Reason: The decision process described in the requirement above ((Re-) configuration Decision) needs to have some kind of input data about the current system state or state of the system environment in the form of information about the system or environment entities that is stored in form of a profile based context model. This requirement is necessary to fulfil objectives II2B and II2G.

Optional Constraint: MexE device classmark 3 features to be supported (i.e. CC/PP over HTTP1.1 and the support of device specific features MIDP as defined for PDA's).

Verification: The scenarios described in 5.2.5 are test cases to verify proper functionality of this requirement. Since this is a demonstrator that shows scenarios, which are based on a middleware concept and not yet fully implemented, it is sufficient to proof that the implementation only works for the demonstration scenario. The implementation does not have to work for other possible scenarios.

5.2.6.2.6 Service Architecture Deployment

Description: This functionality involves handing over service sessions to another provider. This includes handing over to other providers of basic communication services, i.e. switching from GSM to UMTS. Deployment can also be starting new service providers on backup locations.

Reason: The decision about an initial architecture of a service usage or about a necessary reconfiguration, for example to fulfill either changed QoS requirements or existing QoS demands in a changing environment (for a mobile terminal moving from one location to another), should be implemented at runtime

Verification: The scenarios described in 5.2.5 are test cases to verify proper functionality of this requirement. Since this is a demonstrator that shows scenarios, which are based on a middleware concept

and not yet fully implemented, it is sufficient to prove that the implementation only works for the demonstration scenario. The implementation does not have to work for other possible scenarios.

5.2.6.2.7 Profile Signaling

Description: This functionality is concerned with transmitting information gathered by sensors to a consumer that makes use of this information, e.g. for deciding about a reconfiguration. Usually the recorded and post-processed QoS are transferred to the network for delay and bandwidth negotiations in the trading service.

Reason: The origin of profile data can be located or distributed in different locations within the network not necessary in the physical location where the data is used. Therefore there is need for a sophisticated mechanism to signal changes in certain profile elements over the network without adding unnecessary traffic or delay times.

Verification: The scenarios described in 5.2.5 are test cases to verify proper functionality of this requirement. Since this is a demonstrator that shows scenarios, which are based on a middleware concept and not yet fully implemented, it is sufficient to prove that the implementation only works for the demonstration scenario. The implementation does not have to work for other possible scenarios.

However this is a critical requirement regarding evaluation objectives, for example IIA-IIG, so a best effort implementation has to be found.

5.2.6.2.8 Profile Sensor

Description: In these requirements suitable measurable QoS parameters of the running application, e.g. time to completely download of all or a sub-set of objects of an HTML page and streaming bandwidth are defined.

Also monitoring QoS parameters during a session and post-process the measurements (delay, available bandwidth, average value of download time, variance and pdf of transfer and parsing time) are going to be considered.

Reason: This requirement is about getting the necessary profile data that is used for the Reconfiguration requirement described above.

Verification: The scenarios described in 5.2.5 are test cases to verify proper functionality of this requirement. Sensors highly depend on the type of application they are used for. For the demonstrator only sensors will be implemented that are used in the scenarios described in 5.2.5.

5.2.6.2.9 Download

Description: After the user initiated a download either directly by clicking a download link in the web browsing service or indirectly by selecting another service for usage that needs to download data to the user's terminal client (e.g. a GUI download), further interaction should not take place until the download has finished. The download should be robust to varying network quality or temporary loss of connection. If interruptions occur, the download should continue even if the user changes devices, for example if he started downloading a document on the notebook and afterwards switches to a pocket-PC.

Optional: The download service should offer a management interface to the user. With this interface the user can browse pending downloads, starts, pause and stops downloads, redirects them and set a download schedule and changes prioritization.

Optional: The download should use multiple sources if available to increase performance

Reason: Scheduling for downloads is an interesting option to generate additional reconfiguration requests (e.g. changing to UMTS to finish a document download in time). Moreover it is necessary to update basic software in terminals and download sophisticated user interfaces (e.g. java midlets) for location-based services.

Verification: This requirement is verified by a test case that downloads test data to instrumented terminal clients. The downloaded data is then checked for integrity and whether it has arrived on schedule.

5.2.6.2.10 Interruption Handling

Description: In case a wireless link is interrupted during the execution of an application, the middleware should support a persistence mechanism to maintain the current application session state information during the short period of time when the terminal is disconnected from the network

Reason: Wireless Connections of mobile units are unstable by nature. Due to physical reasons the link can be lost any time during an application usage. Applications need to be robust to temporary loss of link (nonfunctional requirement).

Verification: At least one of the implemented scenarios should survive intentional disconnections and must be able to recover to the state of the last application (if the application is still available after reconnection).

5.2.6.3 User Interface Requirements

Within the demonstration, the user interacts with both applications and part of the middleware through the user interface. The following requirements describe user demands that are connected to the user interface but are independent of a specific application or service to use.

5.2.6.3.1 Service Browsing

Description: Using more than one application in a service-based environment generates a need for a generic service access user interface or a service portal. The user connects to the network using a single entry point, comparable to a start- or home page for Internet browsing. There he/she can navigate through a list of applications and services that he/she has subscribed to and are available or reasonable to use at that moment.

Reason: The reason is that in a wireless environment of mobile clients with changing terminal capabilities, while the number of all available services in the whole network might be very large, the number of services that are accessible for one specific user is significantly smaller and moreover might change depending on the terminal and the user location. Also there might be more than one provider for a certain service and while a service might be available all the time it does not need to be offered by the same, maybe local, service provider.

Therefore is not always possible for the user to access services by typing the provider address or using information stored in a bookmark (like in internet browsing). However the user profile may contain information about services he/she has subscribed to or could be interested in or are appropriate to use in a given situation. This information is mapped to the services that are discovered and available to the network at the moment (by discovering their providers).

The user then can browse through the suggested services and select one or more for parallel usage (e.g. watching a video and have an instant messaging notification on screen at the same time).

If one provider becomes unavailable it can be replaced by another offering the same service (taking over the session). The container looks and feel as well as some generic interface elements (e.g. switch off, etc) should stay the same however.

Verification A separately specified and implemented test component, scrapes the main service Browser User Interface and compares the displayed services to its own list of services that should theoretically be discovered in appropriate test cases, i.e. it makes sure the main browser discovers and displays the expected services in each test case.

5.2.6.3.2 Interface adaptation

Description: Service usage is not necessarily bound to a specific device. Within the range of device capabilities a service can accept that the user choose freely an appropriate device, usually this depends on mobility restrictions (see the scenarios described in 5.2.5).

The demonstrated application services should be aware of the current's device capabilities and offer an appropriate interface (e.g. reducing displayed information on small screens).

Reason: The user interface is the most observable part of the service adaptation concept (see Table 5-1), moreover it is connected to all visualization objectives regarding the overall system concept (II1A), all key concept objectives (especially II2A, II2B and II2C) as well as demonstrating customer benefits (II4A, II4B and II4D).

The user interface presentation should be as good as possible for each device, taking into account the current terminal capabilities, because a smooth user interface is necessary for getting non-distorted results that help in proofing the concept objectives regarding concept validation (especially III3A).

Verification: Both demonstrated scenarios (5.2.5.1 and 5.2.5.2) contain a device change. If the user interface of the connected application always fits the screen and contains all necessary controls in orderly and intuitively fashion has been approved by at least four persons without any special technical or IT-knowledge (customer simulation) then this requirement is verified.

5.2.6.4 Application Scenario Functional Requirements

Two major applications will be demonstrated (5.2.5.1 and 5.2.5.2). The requirements for services that are involved in the applications but do not belong to the middleware (i.e. the infrastructure layer) are described in this section.

5.2.6.4.1 Web Service

Description: The user wants a smooth web browsing experience, i.e. browse to a URL without caring about technical details like what type of network currently used, available bandwidth, changing response times and local service providers availability.

The user wants to type a URL or chooses a certain address from a bookmark list. He/she wants to scroll information that is too big to fit on the screen. This scrolling should only be vertical, i.e. paging. He/she wants to click hyperlinks to jump to other information and go back to the page he came from.

The web browsing service is a gateway that connects the user of the current network to the Internet and lets him browse HTML pages with his terminal's browser in a way, which is appropriate for the device capabilities currently used.

Rendering is done on the terminal itself but the service might need to do format conversions (e.g. to WML) to display the HTML pages on the terminal. The service needs to monitor quality of service parameters that are application specific as far as this is possible without modifying the client browser. This data is stored in the service profile and can trigger active (e.g. services asks for higher bandwidth) or passive reconfiguration event (e.g. middleware hands over to another provider).

Optional: The service can have a downloadable client side interface that can automatically be installed on the terminal to support more sophisticated monitoring of client side QoS

Reason: See description of scenario 5.2.5.1.

Verification Scenario 5.2.5.1 also provides a test case that can be used to verify the functional requirements connected to the web browsing service.

5.2.6.4.2 Media Stream Service

Description: The user wants a smooth multimedia experience, i.e. using a dedicated media browser to choose a media file and then watching/listening to the media data without caring about technical details like what type of network currently used, available bandwidth, changing response times and local service providers availability.

The service should monitor quality of service parameters that are application specific as far as this is possible without modifying the client browser. This data is stored in the service profile and can trigger active (e.g. services asks for higher bandwidth) or passive reconfiguration event, for example the middleware handover to another provider with a stream of different quality (resolution, fps etc.) but with same content.

Optional: The service can have a downloadable client side interface that can be automatically installed on the terminal to support more sophisticated monitoring of client side quality of service.

Optional: The web browsing service can be used to search for media files on the Internet.

Reason: See description of scenario 5.2.5.2. Note that this application is only emulated for evaluation purposes, i.e. there will be no corresponding user interface to freely use this service besides one hard-coded test file.

Verification: Scenario 5.2.5.2 also provides a test case that can be used to verify the functional requirements connected to the media service.

5.2.7 Demonstrator Use Cases

In this section the use cases supported by the demonstrator are illustrated. In the demonstrator will focus on two classes of usage types namely:

- I. **Technical Efficiency Evaluation:** the effectiveness of the software or the scalability will be evaluated by measuring software-emulated scenarios.
- II. **Demonstration of the Middleware implementation in wireless environment:** in this usage the usability of the demonstrator will be shown illustrated by a web browse scenario with wireless devices. The purpose of this scenario is to the practical usability of the middleware and the User interfaces.

Each usage translates to a valid workflow that connects several use cases. Some of the use cases can be utilised in both scenarios.

5.2.8 Technical efficiency

A number of users (0..x) are simulated by applying a pre-defined (or pre-recorded) behavior to an example application service (web browsing or streaming-media download) to simulate a productive usage scenario with a significant number of users.

Within this simulated productive usage scenario the behavior of the system regarding one specific test user is monitored and the observations are used to measure performance and evaluation data. This means there is a simulated background load for the system in front of which the systems performance regarding one user will be evaluated. This allows for both evaluation of a single typical use case and the scalability of the system, e.g. how will be the performance for one user influenced if there are too many concurrent users (e.g. 100) compared to few Background users (e.g. 10).

All data necessary for evaluation (see section 5.6) will be logged during the scenario and can be post processed and displayed afterwards. The recorded parameters will be:

- a. The time between an unexpected change in network quality,
- b. The notification of reaching the middleware,
- c. The time between notification and a reconfiguration or adaptation decisions,
- d. The time it takes to realise the decided configuration or adaptation.

With this scenario all main concepts can be demonstrated since it involves finding and using an appropriate service using the service-trading concept. The network reconfigures several times depending on the user profile, for instance if the user wants to have a quick download instead of cheap one. The SADDL deployment, i.e. trading that involves non functional requirements to decide which of the possible Implementations will be returned as an answer to a request for a certain service, is a bit more complicated to involve (and therefore optional), but could be shown by simulating an overloaded component that implements a gateway service (in our scenario a service that allows for browsing of html pages) that could lead to powering up and deploying a second component that implements the same service (i.e. html browsing) on a second server or more simple just load balancing the request from each service client between two parallel running components implementing the same service on two different servers.

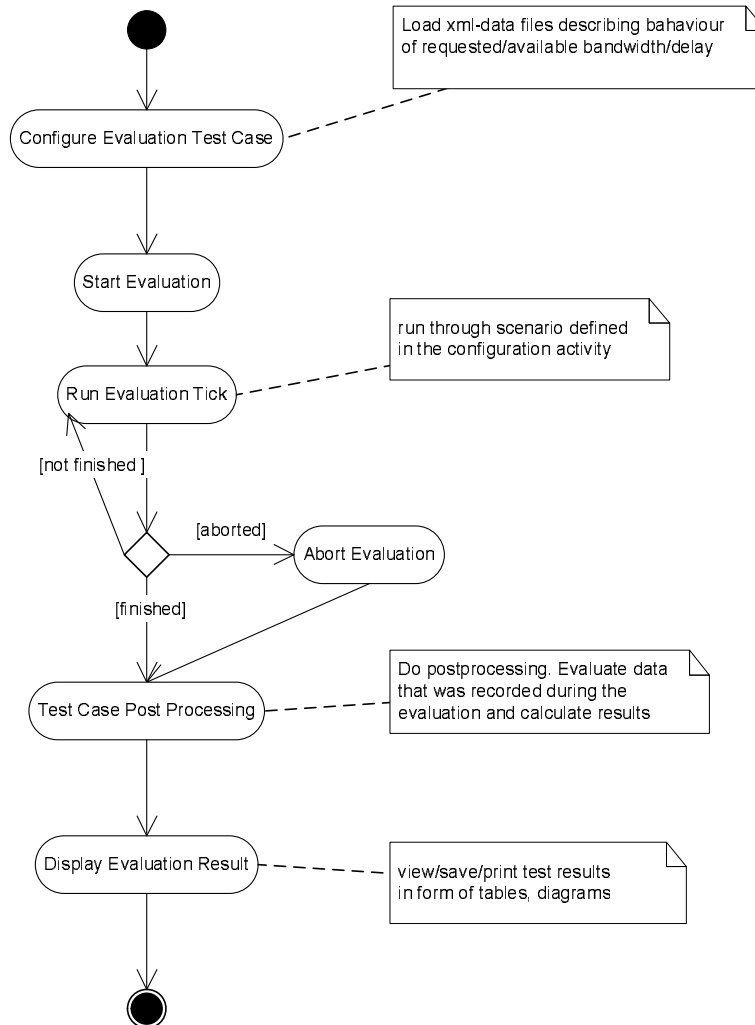


Figure 5-2: Evaluation Activity diagram

5.2.9 Demonstration of Middleware implementation

This usage is for demonstration and presentation purposes. It is a spiced up version of the well-known web browsing application type. The main purpose is to demonstrate that a common mainstream application can be carried out on the middleware infrastructure without any negative side effects for e.g. resource usage and scalability.

The user is connected with a full-featured client (a notebook) to a proxy server that offers an Internet gateway service and he/she can browse any web page he/she likes. After some time a reduction of network bandwidth is emulated. The client notifies the network about this reduction, which let the current service adapt to the new situation by reducing quality of transferred pictures or simply strip the browsed web-pages of all multimedia content. Alternatively the network could handover the browsed session to a

second available internet gateway that is specialized in serving low bandwidth optimized web pages, e.g. by using compressed WML instead of HTML.

Afterward the user switches to another client and can continue browsing the web from the point he left on the notebook (but using a different screen size). He starts a download in a high bandwidth situation. A reduction in bandwidth to 10kbit/s simulates the user leaving the WLAN Hotspot to GSM. The user then decides he needs his download to finish quickly. Therefore the client sends an active request to the network for reconfiguring to higher bandwidth and switching his session to UMTS (emulated by increasing network bandwidth to 384kbit/s).

While the user can use a demonstration application, several use-cases from the evaluation scenario are also active and allow for monitoring user behavior and simulating various interesting conditions (e.g. connection loss, low mode switch etc.) the user can experience "live" while using the application.

On an optional visualisation window, the different service instances (i.e. components implementing a specific service in this case a gateway that can forward html requests like a proxy server) can be monitored in form of boxes that are connect with channels to the clients and exchange messages over these channels. In the GUI the network bandwidth and delay bar as well as notifications sent into the network and the reconfiguration reaction in form of re-wiring communication channels between the service boxes can be monitored (as a log-file). The graphical visualisation functionality is an optional requirement, it can be designed but its implementation is not mandatory.

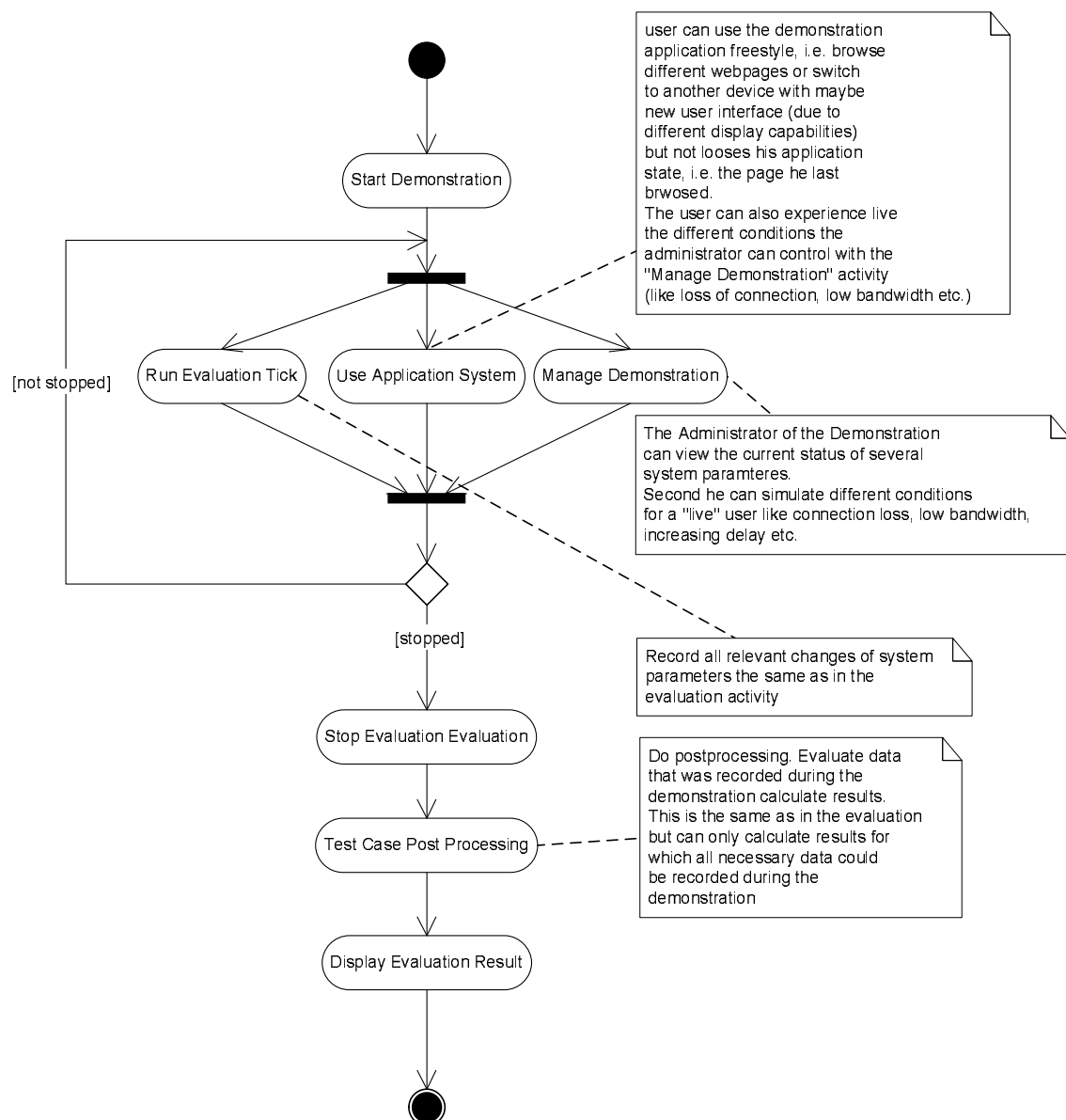


Figure 5-3: Demonstration Activity diagram

5.2.10 Uses Cases Description

In this section a more detailed description of single uses cases that correspond to an activity in the first scenario are given.

5.2.10.1 Web Browsing Use Cases

- **Choosing Application Service**

General comments

The first step in the proposed web-browsing scenario would be the interaction to choose the browsing service as the current application after its availability has been detected. This can be achieved by displaying all available application services on an entry screen that is shown as soon as the user switches on his device. The UIManager manages this screen. The UIManager is an infrastructure service that manages a virtual system access node for the user. Among other things it manages the users selection of application services and device dependent rendering of their user interfaces.

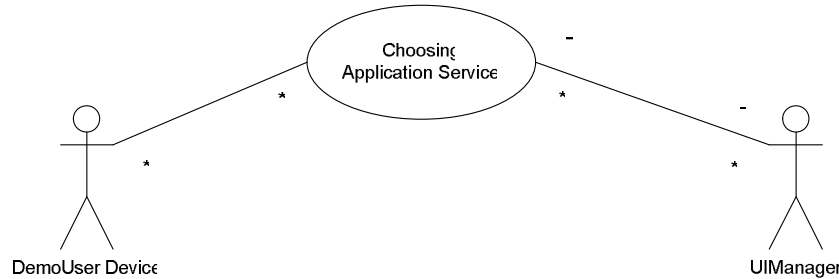


Figure 5-4: Choosing Application Service

Requirements

- Display selectable list of available application services
- Device dependent rendering of a generic user interfaces and combination of different application service interfaces.

- **Browsing**

General comments

After choosing web browsing as the users current application the user interacts with the HTTP browsing service for navigating Internet pages. The UIManager is an intermediary that organises a generic user interface, for example the possibility to switch to another application service or that manages the display of more than one service interface at the same time. An example would be browsing web pages and at the same time get notification about new email.

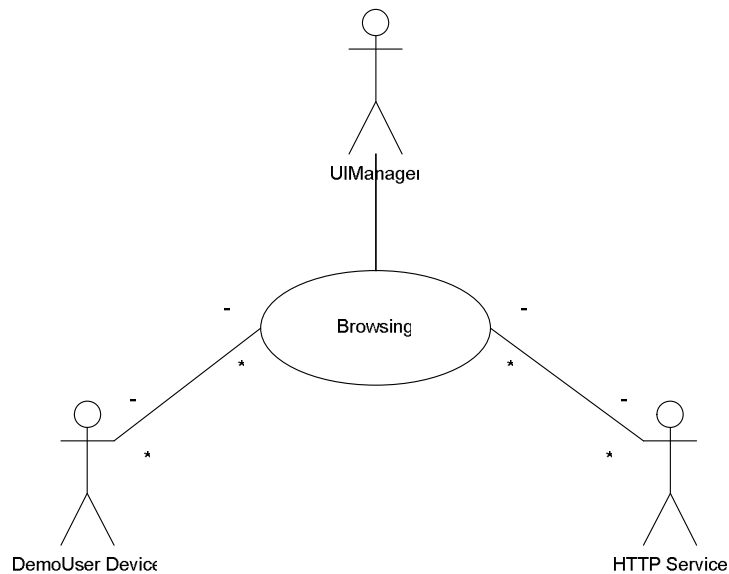


Figure 5-5: Browsing

Requirements

The application service interface is either re-routed and modified through the UIManager, or Completely rendered by the UIManager (optional)

- **Switching between Devices (Optional)**

General comments

The user should be able to have his current session state saved so he can continue his session on another (or the same device) but with different device capabilities, like e.g. larger screen, higher processing power etc.

Requirements

- All characteristics of a current user session need to be saved on a place in the network, so it can be accessed from different devices.
- A session restore or handover need to take probably changed device capabilities into account.

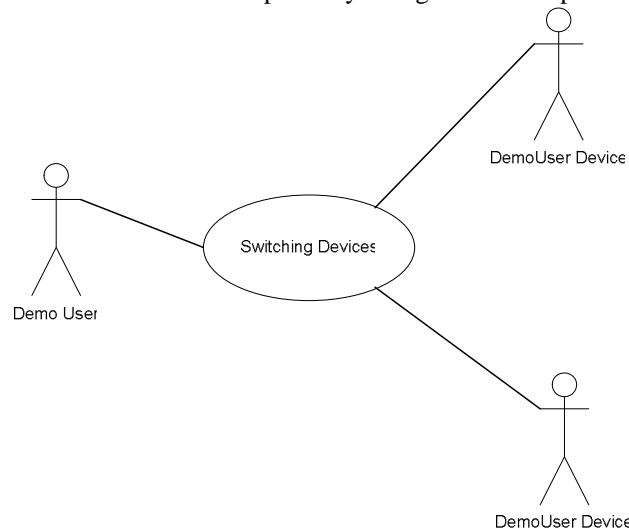


Figure 5-6: Switching between Devices

5.2.10.2 Evaluation Use Cases

In this section more detailed description of single uses cases that correspond to an activity in the second scenario are given:

- **Start Evaluation**

General comments

For evaluation purposes a separate evaluation test user can connect to the demonstrator and start a test run. This test run includes an optional behaviour pattern for both network parameters delay and bandwidth as well as an optional usage pattern and records all relevant system parameters like used and available bandwidth, delay, requested QoS parameter of different applications and reconfiguration decisions together with a millisecond accurate time stamp (see Figure 5-7 for a small example)

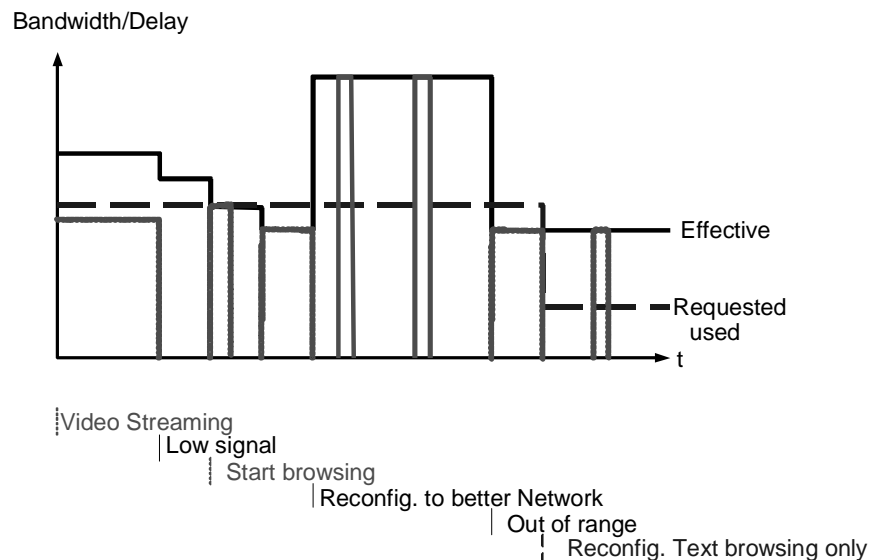


Figure 5-7: Evaluation Test Patterns

Later on from these values a statistics will be generated about timing behaviour and scalability issues of the system.

Requirements

- Possibility to optionally specify network profiles and networks usage test patterns
- Start recording of system behavior and all relevant status parameters including time stamps for later statistical analysis and evaluation.

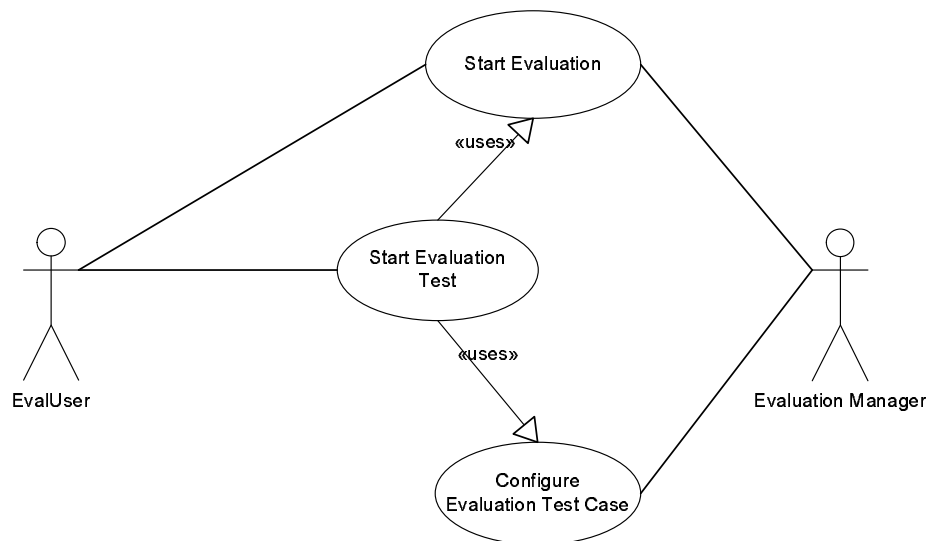


Figure 5-8: Evaluation Use Case

- **Abort Evaluation Test**

General comments

Since evaluation tests can be long or endless running there need to be a possibility to stop a currently running evaluation tests. The so far gathered values however should be stored for later analysis and evaluation.

Requirements

- Menu entry to stop a running evaluation test

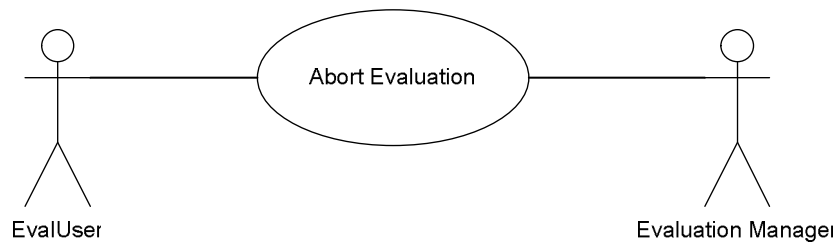


Figure 5-9: Evaluation abort use case

- **Displaying of Evaluation Test Result**

General comments

The data gathered during an evaluation test run need to be statistically analysed and evaluated before displaying it. Expected results are diagrams of several important system characteristics and measurements of critical timing and resource issues. Results of evaluation and analysis should be stored for later access.

Requirements

Following parameters are to be measured and will be displayed to the user:

- Time to detect reconfiguration necessity (average, min, and max values)
- Time for reconfiguration/adaptation decision (average, min, and max values)
- Time for reconfiguration deployment, or to adapt (average, min, and max values)
- Maximum possible reconfig. freq (Minimum time between two non overlapping reconfigurations).
- Average time for accessing current profile (one data element) value
- Average time for accessing of history profile(one data element) value

The following scalability values are to be measured:

- Average memory usage for storing one profile data element
- Average size & memory usage for profile per user
 - Average size & memory usage for profile history per user per day
 - Average size of network profile
 - Average size of terminal profile
 - Average size of service profile
 - Dependency between timing value and number of concurrent users (optional)

The following diagrams are to be presented:

- Available bandwidth over time
- Requested bandwidth over time per service
- Network delay over time
- Maximum tolerated delay per service
- Dependency between timing value and number of concurrent users (optional)

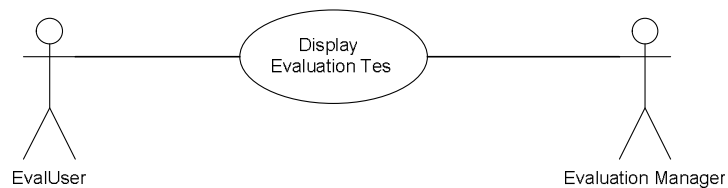


Figure 5-10: Displaying of Evaluation test results

- **Display System Status**

General comments

It is necessary for the evaluation purposes that the current system status parameters, either for debugging or presentation purpose to be monitored. The status display should have a clean design, so it can be used in presentations. In such cases graphical display is recommended.

Requirements

The display should contain the following information:

- Structure map of the demonstrator (box diagram)
- Visual indicators for important events, such as:
 - Detection of reconfiguration necessity.
 - Reconfiguration or adaptation decision
 - Accomplishment of the Reconfiguration
- Visual representations of the following data:
 - Type and overall capability of the device
 - Available bandwidth
 - Used service
 - Available services

There needs to be a (preferably web based) GUI to select the type of display.

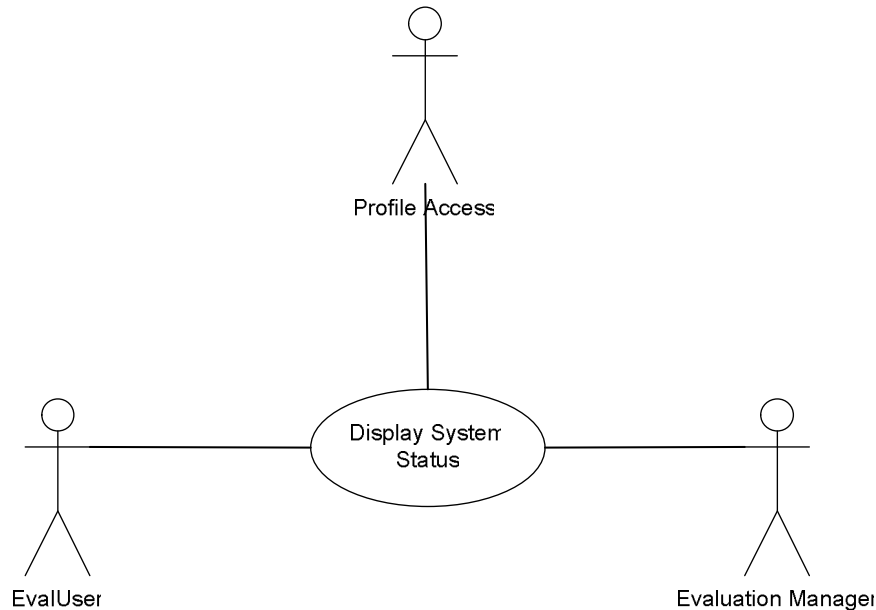


Figure 5-11: Display of the system status

- **Manual Profile Manipulation**

General comments

It is possible that the current system status parameters can be changed online, either for debugging or presentation purposes.

Requirements

The parameters that need to be modified are:

- Available bandwidth over time (bandwidth vs. time behaviour),
- Network delay,
- Availability of services over time,
- User preferences at different bandwidths for chosen application.

There need to be a (preferably web based) GUI to manipulate the parameters online.

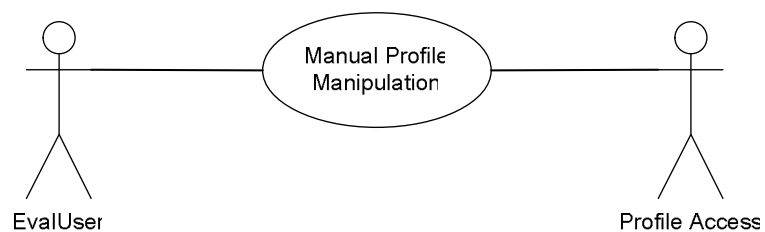


Figure 5-12: Manual profile manipulation**5.2.10.3 Combined Use Cases**

The third group of use cases contains a single case that exists as a mixture of usability and technical efficiency.

- **Combination Use Case (Optional)**

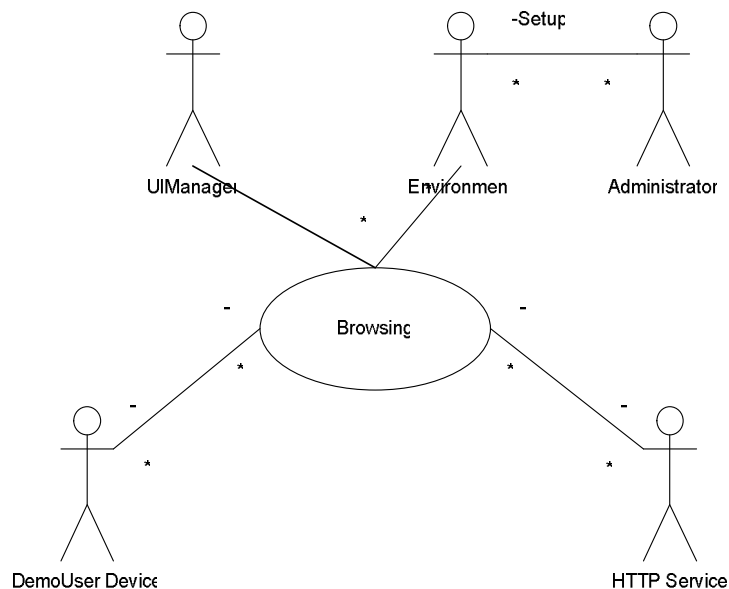
General comments

A mixture of usability and evaluation scenarios can be presented in this use case. The idea is to experience a varying number of concurrent users during a usage scenario.

After choosing web browsing as a current application, the user interacts with the HTTP browsing service for navigating Internet pages. The environment simulates heavy usage by the following attributes:

- Amount of users,
- Amount of services,
- Delay time,
- Bandwidth.

The UIManager is an intermediary that organizes a generic user interface, for example the possibility to switch to another application service or that manages the display of more than one service interface at the same time. An example would be browsing web pages and at the same time get notification about new email.

**Figure 5-13: Browsing****Requirements**

- The application service interface is either re-routed and modified through the UIManager, or
- Completely rendered by the UIManager (optional)

5.2.11 User interface

The following three figures show simple GUI prototypes of the three main screens regarding the demonstrator administration and controlling. The ControlCenter is the main screen that shows the current status of the demonstrator and allows for start and abortion of test scenarios. During live demonstrations only it also allows for manual manipulation of several system parameters to force several QoS contions that might lead to a reconfiguration, for example by decreasing the effective bandwidth.

ControlCenter GUI Prototype:

Testmode: Freestyle (not running)

Effective Bandwidth: 9.6kb/s 14.4kb/s 56kb/s 144kb/s kb/s

Effective Delay: 5ms 10ms 50ms 100ms ms

Max. Bandwidth: 9.6kb/s 14.4kb/s 56kb/s 144kb/s kb/s

Background Load Sim.: 5user 10user 100user 1000user user

Test Client Probes

Client	Eff. Bandwidth	Total Delay	Service used	Service QoS
Test1	14.4	50	html full	2

System Monitor

Users: 1	Avg. Reconfig Time: 10ms	Avg. Reconfigs/min: 0.2	Number of Services: 2
----------	--------------------------	-------------------------	-----------------------

Figure 5-14: Live Demo and Evaluation Control Center

TestCenter GUI Prototype:

Test Scenario Configuration

Type of Test: Freestyle Script

Test Script:

Test Result Set:

Iterations:

Runtime:

Figure 5-15: Configuration of Test Scenario

In the TestCenter GUI the exact nature of an evaluation can be configured. Freestyle Test means live demonstration of an application using real hardware terminals. An evaluation usage can be configured using a script that defines the exact test parameters like number of users, their access behavior and the network behavior.

EvaluationCenter GUI Prototype:

Test Result Evaluation

Type of Evaluation: Short Long

Test Result Set:

Evaluation Result Report:

Figure 5-16: Evaluation Result Postprocessing Control

In the EvaluationCenter GUI test results can be reviewed and any type of post processing can be configured and applied.

5.3 Design Specification Overview

5.3.1 XML based data storage and retrieval

Communication profiles will be described in XML. A registry that is a small database is used to store and manage the *communication profile* on the according server or device. Communication profiles contain information and settings for all the hardware, software, users' preferences, and terminal capabilities, divided into the according profiles of the entities: terminal, user, network and services.

5.3.2 Main Demonstrator Function Groups

In the previous document the requirements for the main demonstrator function groups have been identified. In this section a brief overview about the function group is given. Detailed description of the specification of the function groups is given in chapter 5.5. The main demonstrator function groups are:

- **Applications:**

A web browsing application will be implemented to demonstrate the most import concepts of the middleware technology.

- **Terminal Management:**

The demonstrator should support different types of end devices hence a terminal manager is needed that can handle different cases.

- **Simulation:**

A tool, which simulates different QoS, will be developed (adjustable bandwidth & delay).

- **Trading:**

Different service should be offered and selected by a trading service depending on complex preferences (see profiles)

- **Reconfiguration:**

Services should be reconfigurable such that adaptation to changes in environment will meet user expectations and system requirements, in otherwords while using a service it should be possible to switch to better-suited service.

- **Evaluation:**

One of the aims of the demonstrator is to analyse the behaviour of the system during the reconfiguration process. To evaluate the process, values of the effective delay (static delay & delay given by bandwidth) will be measured.

- **Profiles:**

To realise the trading manager as well as the reconfiguration controller e communication profiles will be used to describe all relevant actors (user, network, services, terminal) of a mobile scenario and hence the context (state of system environment) the system is in. This context information is used by the trading and reconfiguration processes as input for decision making.

5.3.2.1 General Overview

Figure 5-17 gives a general overview about the components of the demonstrator and their dependencies.

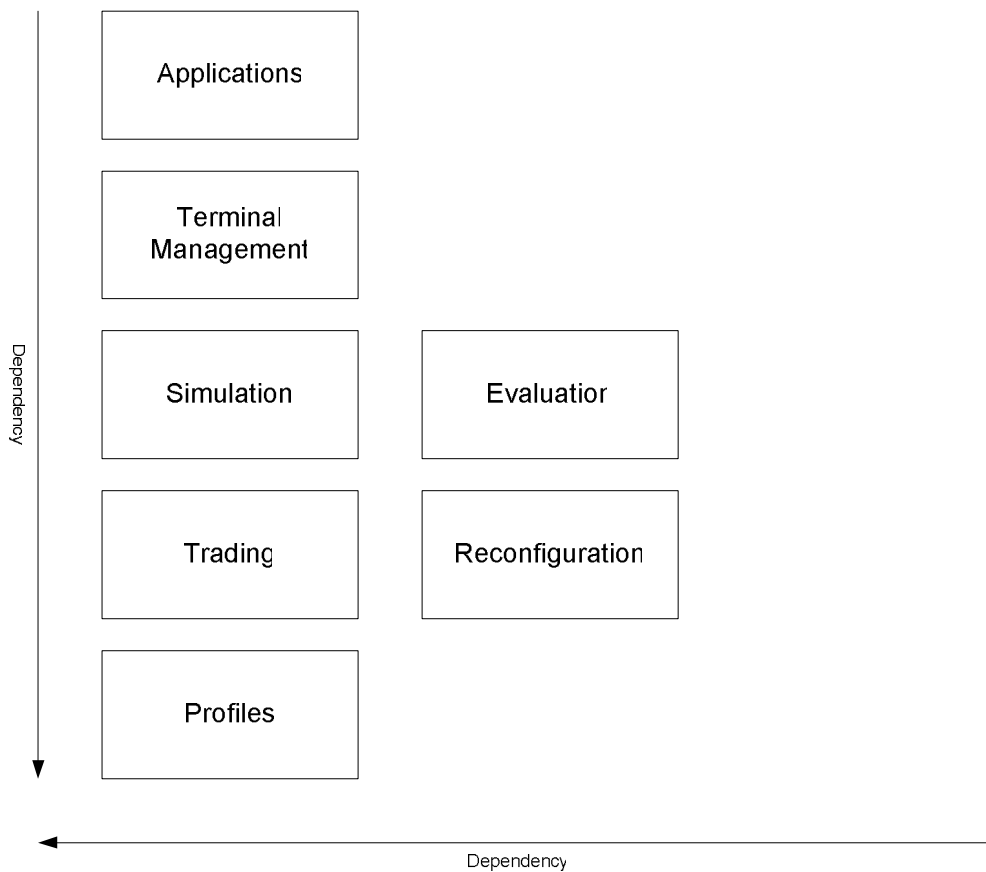


Figure 5-17: Demonstrator components

5.3.3 Main Deployment Locations Description

Interactions between terminal and network are crucial, as the available bandwidth on the wireless link is a limited resource that should be used for services rather than negotiations. Furthermore, resources on the terminal itself are also limited, in order to relieve the terminal from the burden of frequent interactions with network entities, the functional architecture for network supporting reconfigurable terminal [5-1] has been established in which three proxies have been specified, Proxy Reconfiguration Manager (PRM), Serving Reconfiguration Manager (SRM) and Home Reconfigurable Manager (HRM).

For our demonstrator it is absolutely enough to use only a PRM and a HRM to show the general functionality. Furthermore we will not implement an application for the client. Our main application is a normal web browser. Figure 5-18 shows the three location of the system. In the demonstrator the terminal will not have any functional part. We use a terminal agent on the proxy, which evaluate the terminal functionality.

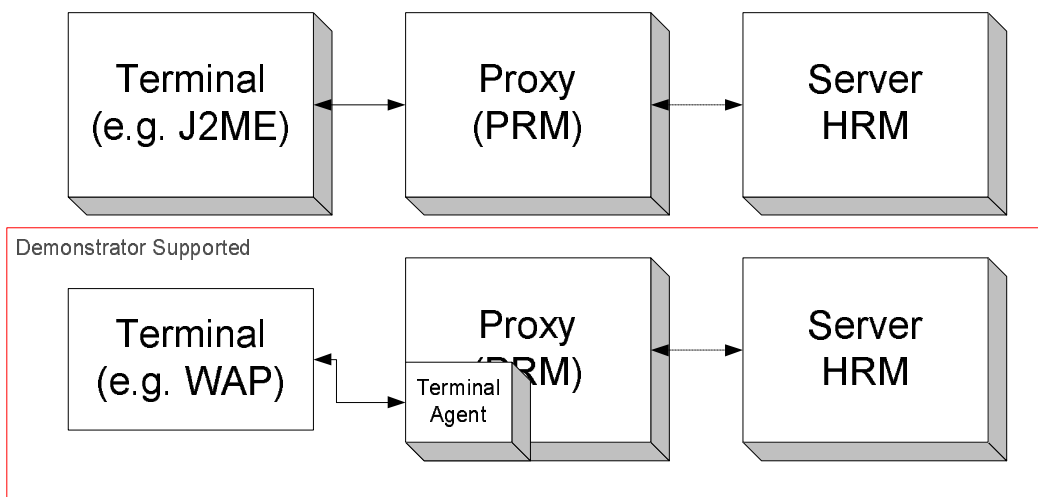


Figure 5-18: Demonstration scenario

In Figure 5-18 demonstration scenario with two different types of terminals is illustrated. The first terminal type is one terminal that can download software (e.g. have a J2ME), the other one is a terminal that is not capable to download and execute client side program code, like most today's WAP phones.

For the demonstrator development and the presented scenarios only the second type of terminals will be used. This means that all implemented end user scenarios will be comparable to using a WML or HTML browser (no download of code to the client). All software (like a downloadable user interface) will be downloaded to a kind of proxy for the terminal that runs on the fixed network (this is what is called "Terminal Agent" in Figure 5-18)

The advantage of this kind of scenarios is that any terminal type with at least the capability for WML Browsing can be used without any changes to the demonstrator implementation.

The main type of terminal in the demonstrator is however a FSC LooX pocket PC. This kind of terminal can have a J2ME virtual machine installed on it and therefore could also support extensions of the demonstrator that would need a terminal of the first capability group. This is for later extensions of the demonstrator or integration with other developed demonstrations within the SCOUT project but will not be used in the evaluation and presentation scenarios described in this document.

Besides the PocketPC other types of terminals can be used in the presentation scenarios, e.g. a WAP capable GPRS mobile phone with maybe slightly reduced functionality depending on the capabilities of the used terminal. This is not necessarily valid for the evaluation scenarios because during the evaluation QoS needs to be directly measured on the terminal. This requires the capability to run client side code (Java script) on the client. Most terminals should support this without modifications if they support HTML browsing but without this capability running the evaluation scenario or any combination of presentation and evaluation (e.g. presenting with simulated concurrent users in the background) is not possible.

For the purposes of the simulator the HRM is just another instance of the PRM functionality that can be optionally installed to simulate another network. One of the installed network PRMs is then marked as "home" and holds most of the static user profile, so PRMs need to be aware of the fact that there are other networks and that they can request the static user profile if they do not have a cache copy of it from the PRM that is marked as home. Usually the HRM could be different for different users but to ease demonstrator development there is only one home PRM=HRM for all users.

5.4 Base Technology Overview

In this section we describe briefly which technologies we decided to use for the implementation of the demonstrator.

5.4.1 Differences between Services and Components

Three conception problems of today's emerging pervasive network infrastructures such as the Internet and wireless networks (e.g. GPRS, UMTS and HiperLANs) are addressed by services. In the following the three problems are identified:

I. Availability of distributed functionality:

Distributed functionality hosted by 3rd parties within wireless networks and Internet is subject to changes during the runtime of an application. For example the content of the www is changed if a web page move from one server to another, become temporary not reachable or is deleted, but the same information is available on other pages, the content of pages is updated or new pages are published.

Applications that are aim to run in such kind of distributed functionality need to have additional flexibility built in, e.g. the functionality that is to be consumed within an application need to be mapped to a valid source for this functionality during the runtime of the application.

Since a source of functionality and its structure can be distributed itself, not only the actual implementation of a used functionality might change, but also their architecture.

In other domains similar reasons can be found why a flexibility in architecture or implementation during the runtime of an application is either necessary (e.g. due to mobility and unreliability) or advantageous (e.g. update long running or important applications without restarting).

Therefore services should allow for specification of applications with parts of architecture or implementation changing at runtime by separating functional dependencies, their distribution and implementation.

II. Heterogeneity.

It is impossible to standardise the way distributed functionality within a system is accessed for any possible functionality. So two 3rd party providers of the same functionality might have very likely defined two different ways to access it. This can be minor differences in the syntax of accepted messages (e.g. changed order of two parameters) or major differences even in the observable behaviour of an access protocol like the order of two consecutive messages. The provided functionality (e.g. booking a flight) can be the same though and there might exist even adapter functionality from a 3rd provider that translates between both access protocols and the protocol of the application that needs this functionality.

In such cases a search for functionality to be bound at runtime cannot be based only on syntactic interfaces or behaviour specification (regarding an interface), but need to be based on an abstract description of the task that should be accomplished. A subsequent search for example can then look for translating functionality between the two mismatching access interfaces. Therefore services need to allow for separation of provided or consumed functionality and technical details like the access protocol.

III. Distributed wide-area networks:

In distributed wide-area network certain characteristics of sources of functionality can dynamically change, depending on their environment, even if they use the same implementation and therefore specified behaviour, for example if the same code is deployed in different parts of the network with one part experiencing congestion problems and therefore influences the response time of one source of functionality whereas the other is unaffected.

Therefore instances of a service need to be distinguishable even if they share the same implementation and part of the specification of a service instance need to be changeable during deployment or even runtime phase.

5.4.2 .Net

For Implementation we decided to use the new .NET framework by Microsoft. The reason is that .net seems for us to be the most reliable and comfortable development environment for the needed core technologies. Due to the strict and limited resources and project plan we need a development environment that automates as much as possible of the web service code generation and allows a rapid prototyping process. In the following we present some of the core technologies (as web services and XML) and their integration into .net.

5.4.3 Web services

Web services are reusable units of functionality that are dynamically allocated and accessed via the Internet. We use the web service approach, with its infrastructure like the UDDI (Universal Description Discovery and Integration) and its standards like WSDL (Web Service Description Language) as a basis we plan to extend. Although we use .net as an development environment, Web services and its infrastructure are not a proprietary technology but a standard that is approved by the W3C. So the developed system is open and not subject to any technological property.

5.4.4 C#

As implementation language we decided to use C#. The reason for this is that C# is easy to learn (very similar to Java) and fully integrated into the .net framework. Nevertheless, due to the .net framework all developed classes and components can be used by and .net enabled language environment, such as Visual Basic, C++ Jscript etc.

5.4.5 XML

XML is a mark-up language for documents containing structured information. Structured information contains both content and some indication of what role those content plays. Almost all documents have some structure. A mark-up language is a mechanism to identify structures in a document. The XML specification defines a standard way to add mark-up to documents.

In HTML for example, both the tag semantics and the tag set are fixed. An `<h1>` is always a first level heading and the tag `<ati.product.code>` is meaningless. The W3C, in conjunction with browser vendors and the WWW community, is constantly working to extend the definition of HTML to allow new tags to keep pace with changing technology and to bring variations in presentation (style sheets) to the Web. However, these changes are always rigidly confined by what the browser vendors have implemented and by the fact that backward compatibility is paramount. And for people who want to disseminate information widely, features supported by only the latest releases of Netscape and Internet Explorer are not useful.

XML specifies neither semantics nor a tag set. In fact XML is really a meta-language for describing mark-up languages. In other words, XML provides a facility to define tags and the structural relationships between them. Since there's no predefined tag set, there can't be any preconceived semantics. All of the semantics of an XML document will either be defined by the applications that process them or by style sheets.

5.4.5.1 XSLT

XSLT is designed for use as part of XSL, which is a style sheet language for XML. In addition to XSLT, XSL includes an XML vocabulary for specifying formatting. XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary.

XSLT is also designed to be used independently of XSL. However, XSLT is not intended as a completely general-purpose XML transformation language. Rather it is designed primarily for the kinds of transformations that are needed when XSLT is used as part of XSL.

5.4.5.2 XPath

XPath is the result of an effort to provide a common syntax and semantics for functionality shared between XSL Transformations and X-Pointer. The primary purpose of XPath is to address parts of an XML document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and Boolean. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document.

5.4.5.3 XQuery

As increasing amounts of information are stored, exchanged, and presented using XML, the ability to intelligently query XML data sources becomes increasingly important. One of the great strengths of XML is its flexibility in representing many different kinds of information from diverse sources. To exploit this flexibility, an XML query language must provide features for retrieving and interpreting information from these diverse sources.

XQuery is designed to meet the requirements identified by the W3C XML Query Working Group. It is designed to be a language in which queries are concise and easily understood. It is also flexible enough to query a broad spectrum of XML information sources, including both databases and documents.

5.5 Specifications of the Main System Components

5.5.1 Web Browsing Application

One service/application for demonstrating the key concepts of the middleware is being selected. With the Web Browsing service it is possible to show/demonstrate the functionality of a service adapted to different networks and environments.

The user wants a smooth web browsing experience, i.e. browse to a URL without caring about technical details like what type of network currently used available bandwidth, changing response times and local service providers' availability.

The user wants to type a URL or chooses a certain address from a bookmark list. He/she wants to scroll information that is too big to fit on the screen. This scrolling should only be vertical, i.e. paging. He/she wants to click hyperlinks to jump to other information and go back to the page he came from.

The web browsing service is a gateway that connects the user of the current network to the Internet and lets him browse HTML pages with his terminal's browser in a way, which is appropriate for the device capabilities currently, used.

Rendering is done on the terminal itself but the service might need to do format conversions (e.g. to WML) to display the HTML pages on the terminal.

5.5.2 Terminal Management

Service usage is not necessarily bound to a specific device. Within the range of the device capabilities a service can accept that the user choose freely appropriate device, this is usually depend on the mobility restrictions.

The demonstrated application/services should be aware of the current's device capabilities and offer an appropriate interface (e.g. reducing displayed information on small screens).

The user interface is the most observable part of the service adaptation concept, moreover it is connected to all visualization objectives regarding the overall system concept, and all key concept objectives as well as demonstrating customer benefits.

5.5.3 Simulation

For demonstration purposes, a wireless LAN is used for showing most of the middleware concepts and to measure certain parameters. However, it is also of interest certain parameters of networks with lower bandwidth (i.e. GSM, GPRS...) to be measured and displayed. Therefore different bandwidth will be simulated i.e. to reduce the bandwidth by for example resent the data twice. Based on the bandwidth reduction the behavior of the system and forced reconfiguration are going to be analyzed. Figure 5-28 shows how the simulator is integrated into the system architecture.

Any change in bandwidth is simulated on the application layer by multiplexing additional dummy data into the data stream between the client terminal and the terminal management component that offers the generic portal like user interface to several other application services. This means the terminal is always connected to this component that acts like a proxy service. Within this proxy a simulation plugin muxes the stream between the terminal browser and the any application service on the level of the http protocol. This is possible because the two possible application types in this demonstrator are both based on the http protocol. This mechanism does not work for application services that are not based on http. The additional dummy data muxed into the stream is handled by the http protocol as additional protocol overhead that means transparently reducing the effective available bandwidth for the application. As mentioned this simulation based on multiplexing dummy data into the protocol stream on the application layer is rather simple for the two given scenarios that are both based on the http protocol for exchanging html or xml information. Both data formats are based on an extensible markup language, which means that any application that parses this kind of data format ignores new or unknown tags. This way it is possible to add a new tag that is filled with random numbers. This tag and its content are transferred but then automatically ignored by the application receiving the data.

The bandwidth simulation tags payload should be filled with random data, so there is no interference with data compression that might be possibly used on lower layers of the protocol stack.

This bandwidth simulation will only work for scenarios that are based on the http protocol and transport HTML or XML based data.

5.5.4 Evaluation

The evaluation of the whole middleware concerns certain data gathering that allows for measuring certain system characteristics, which are crucial to make statements about the usability of the total system (i.e. the middleware and possible applications).

Most performance values gained in the evaluation are based on the measurement of two basic QoS values namely effective bandwidth and effective delay. Both can be measured quite simple for the evaluation usage of the demonstrator since there is no real hardware terminal involved. To simulate a large number of concurrent users and to have comprehensible results. A user terminal is simulated by a software component that can uses services that are based on the http protocol. Effective bandwidth can be measured after calculating total size of a html page and transmission time. Effective delay can be measured as a round trip between terminal and terminal management proxy. Doing the same for the live experience usage of the demonstrator that involves a real hardware terminal device is more difficult since there is no easy way to patch the existing webbrowser on the pocketPC and writing a browser from scratch is out of scope of the demonstrator. Besides that each modified browser would only support one specific platform. Therefore in this case the transmitted html code is transparently modified with client side executable javascript code that should be understood by most higher capability devices. This code measures bandwidth and delay and sends the result back to the terminal management proxy.

For higher level application QoS only one type of measurement is supported. This measures the time until the complete text of a html page is displayed, the time until the first picture is displayed, 50% and 100% of the pictures. The technology used is the same as for measuring effective bandwidth/delay, i.e. a software component simulation a terminal in the evaluation usage and client side java script code signaling for the presentation usage on real terminals.

A better application QoS would be for example the time until the first page that fits on the screen of the current terminal is completely loaded and displayed. This however would involve complicated rendering algorithms based on varying screen sizes. This is however out of scope of the resources available for implementation of the demonstrator.

5.5.5 Trading

Application/services will expect the middleware to act as an authority that they can consult which will deliver the partner details directly or list the possible candidates. It will act as a simple directory, delivering just the set of intermediates that might finally direct the client to the requested service with an implied negotiation between the parties who offer the services and using it, taking into account quality of

service of the traded functionality as described in the following requirements for (Re-) configuration Decisions below.

5.5.5.1 Polices and strategies for trading

What plays a role in trading is yet not clearly defined, but from the users point of view trading delay is the most important point for decision. Users cannot tolerate long delays and want to access/use the service in less time as possible. However delay is composite parameter (static & dynamic), it is possible that to alter the static delay for high priority user (e.g. divert the traffic to less loaded server). On the other hand, the application point of view in trading is based on the bandwidth. Therefore policies/strategies for trading should be a combination set of both views.

5.5.5.2 Trading Manager

How the trading manager works is described in Figure 5-19. The trading manager selects the best service depending on the communication profiles [5-5]. After the client sends a service request to the trading manager a profile lookup procedure will be started. In case of mismatch or unfound service the trading manager will try to decompose the requested service and compared with the existing services such that a more successful process in finding a suitable service will result. The trading manager has also to check the interactions between the selected services and co-ordinate the communication. Finally the client gets back a service reference or a service stub, which is needed to use the service.

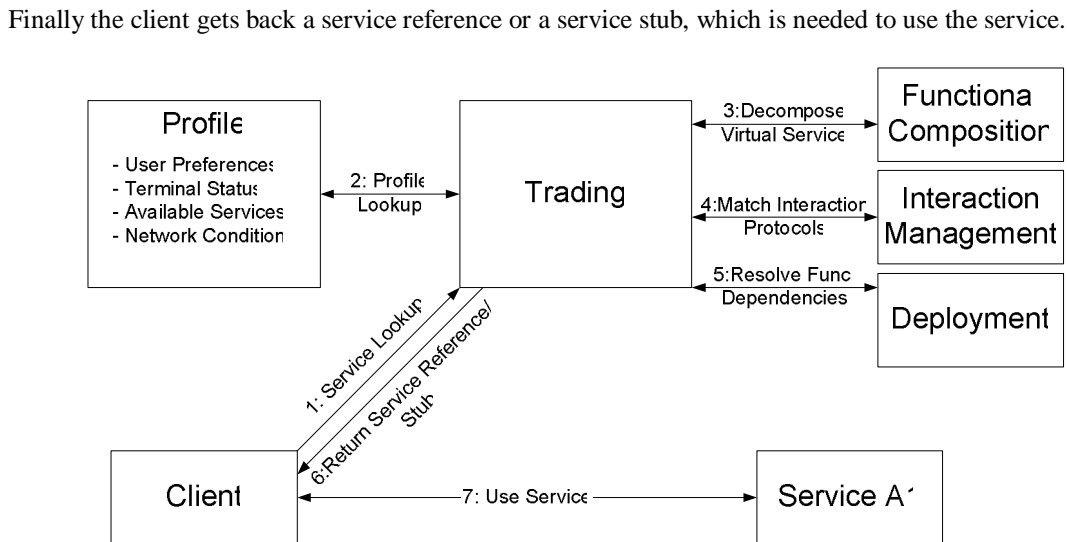


Figure 5-19: Trading Manger

5.5.5.3 Trading Protocols

Two different cases are distinguished, first the client initiated trading that is used when the client is requesting the trading manager to resolve a new functional dependency (shown in the first two diagrams).

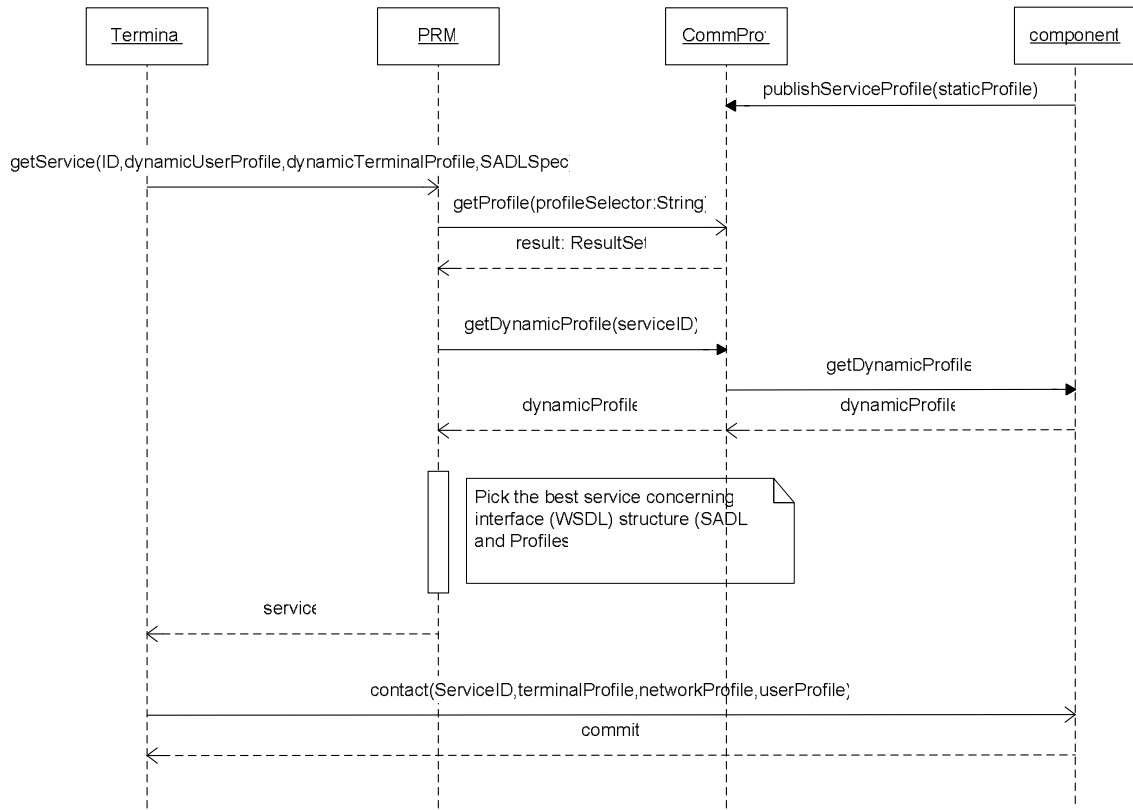


Figure 5-20: Trading Protocol, Case one

In the second diagram the reverse process (unbinding a functional dependency) is shown

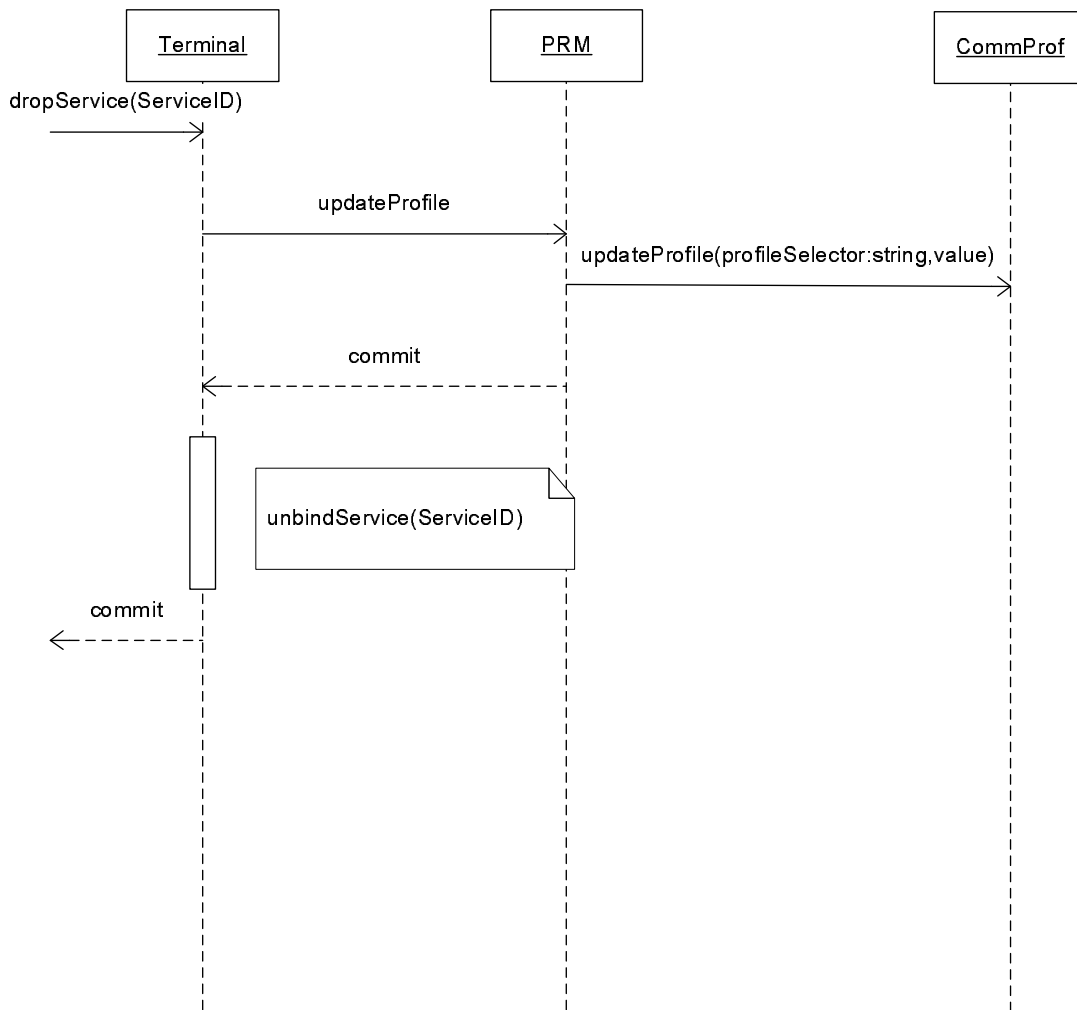


Figure 5-21: Trading Protocol, reverse process

The third diagram shows an example for the second case, i.e. a system initiated change of service bindings. In this case a value in the network profile changed (bandwidth) and is not longer in the agreed range for the currently used service. Therefore the system reconfigures to a new service that can operate in the new network situation. The middleware signals the client to replace the currently used service with the new one.

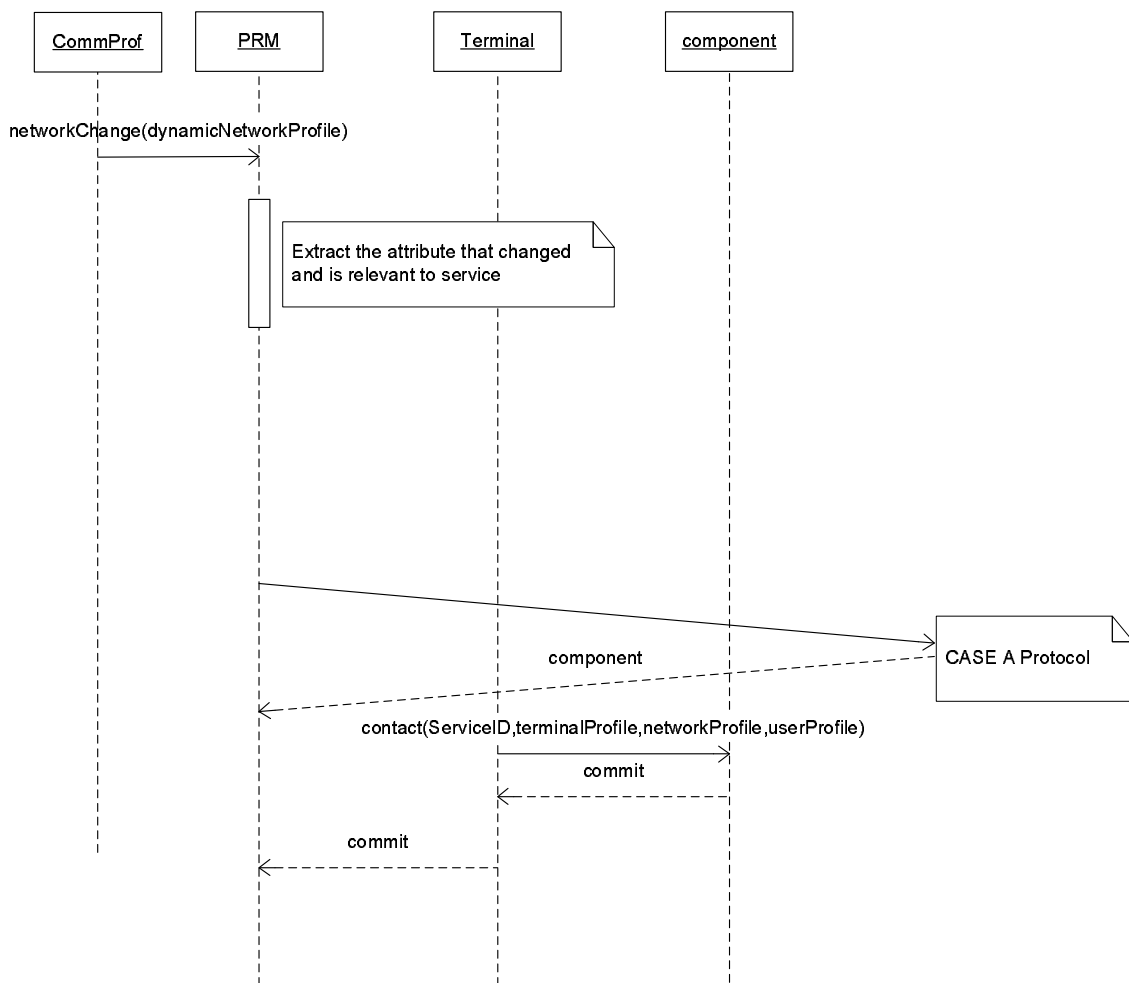


Figure 5-22: Trading Protocol, Case two

5.5.6 Application Adaptation (Optional)

Since middleware-initiated reconfiguration might be time-consuming, service-level adaptation attempts to introduce an intermediate step that gives a service/application a chance to react to changes in the environment (such as the network) that would otherwise cause a reconfiguration.

5.5.6.1 Sample Scenario

A sample scenario could involve the following interaction between middleware and application components:

- An application will be launched and provided with user profile information.
- The application facilitates the middleware for composing structurally complete systems.
- The application registers with the middleware for general events requiring corrective actions, for dedicated notifications with respect to specific changes, or a combination of both.
- Changes detected by the middleware (in this case changes of bandwidth and delay) would be communicated to the application component registered.
- If the application can tolerate or address such changes, possibly by interacting with their service components, the middleware remains inactive with respect to those changes.
- If the application is unable to either tolerate such changes, or to adapt to them, the middleware performs corrective actions as proposed.
- If neither application-level service adaptation nor middleware-initiated application reconfigurations are applicable, affected applications might be terminated.

Addressing application-level service adaptation requires the provisioning of additional APIs between middleware and application, as well as between the client of a service and a component offering this service.

The following sequence diagram (Figure 5-23) illustrates a service-level adaptation request that fails. This failure leads to a system reconfiguration.

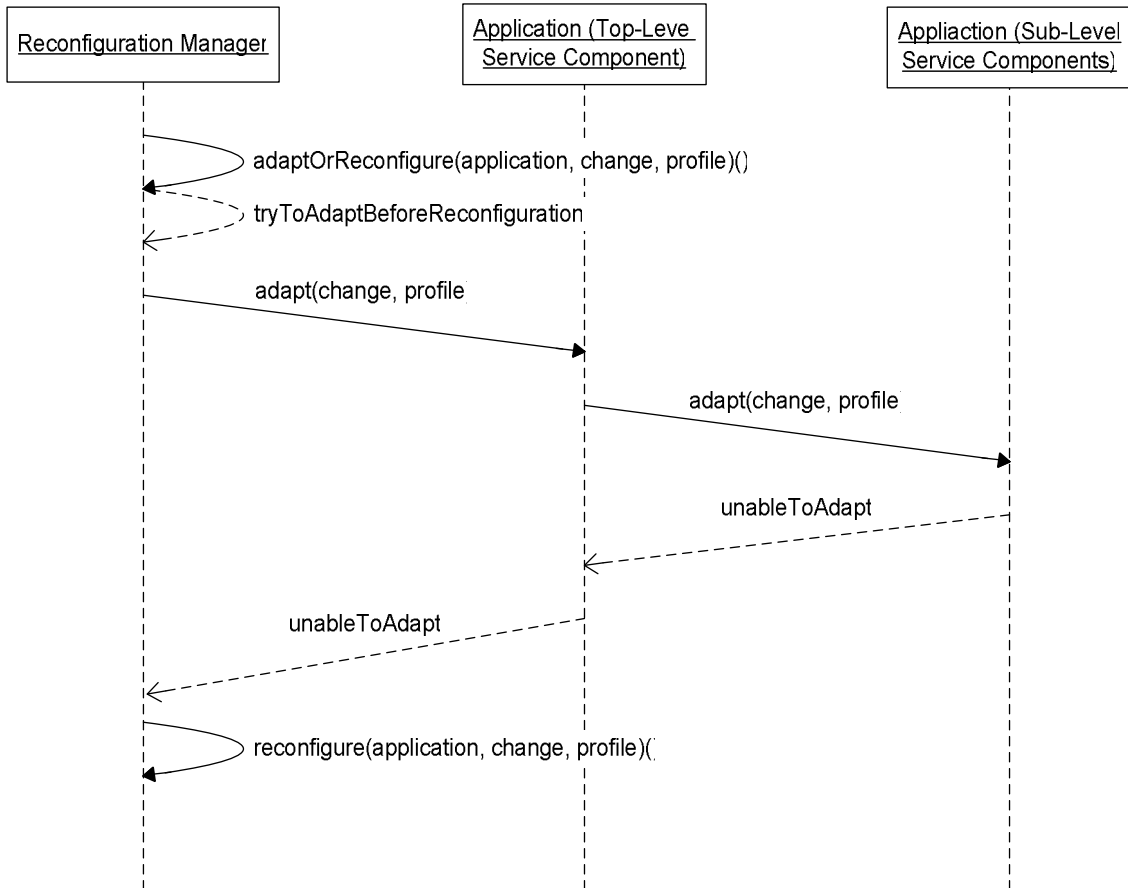


Figure 5-23: Failed Adaptation

The following sequence diagram (Figure 5-24) shows a successful service-level adaptation request that makes system reconfiguration unnecessary.

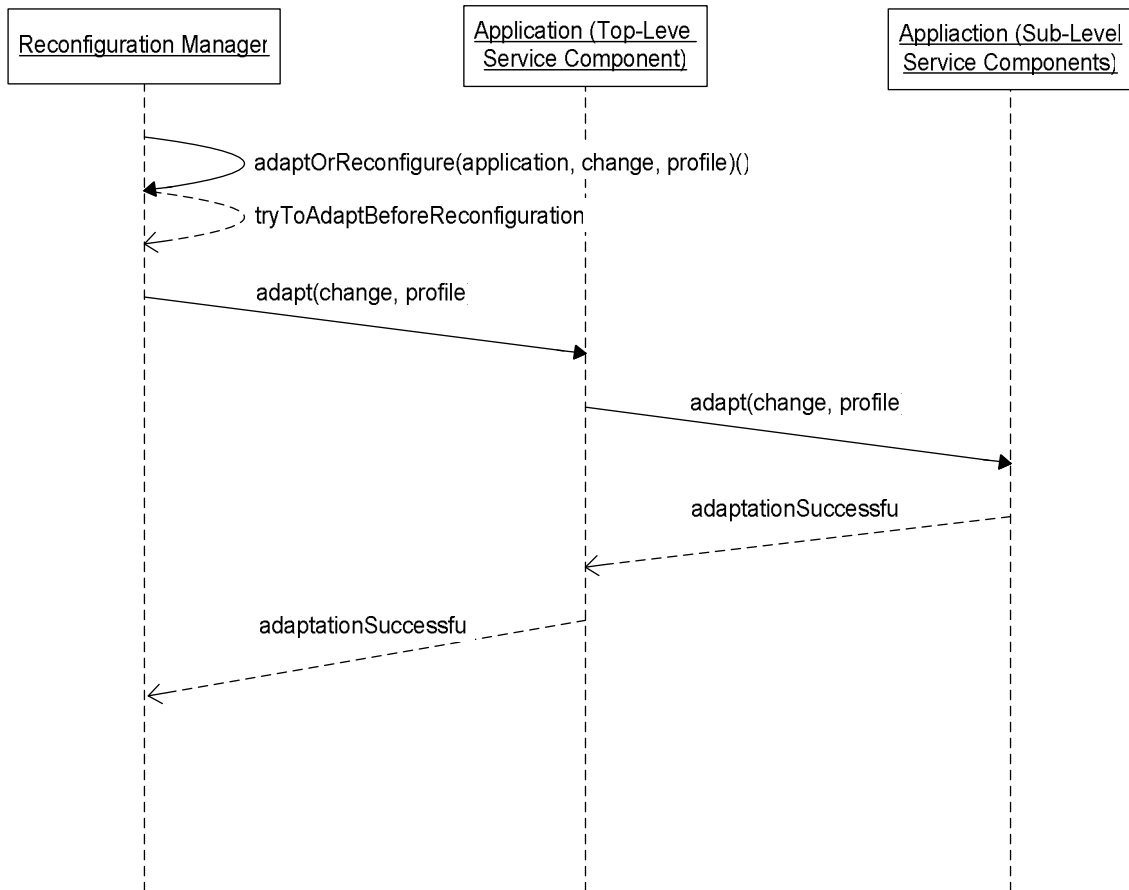


Figure 5-24: Successful Adaptation

5.5.6.2 Measuring Adaptation Time

The time to be measured for adaptation is the time between

- The issuing of an adaptation request from the reconfiguration manager (controller?) to an application or service component, and
- The completion of that adaptation request.

The following results will be distinguished for completion of an adaptation request:

- Adaptation successful, and
- Unable to adapt.

5.5.7 Reconfiguration

There is a need for an automatic decision unit that monitors all Quality of Service parameters, which are involved in an actual situation of a service-based system. By mapping the user profile/ preferences and the application profiles of delay and bandwidth terms into the present Network QoS Parameters, conclusions can be drawn about the necessary dynamic requirements (e.g. when low bandwidth I do not need pictures in HTML file, but only a short text description of suppressed pictures) and initiate their implementation by reconfiguration.

The reason is that not all-possible requirements in applications lifetime can be implemented by simple superposition in a static system structure, especially if resource constraints within the system exist. For example it might be technically possible to construct terminals with three different built in radio interfaces, but 6 built in interfaces might not be reasonable any more regarding complexity.

Reconfiguration can help to solve this problem since it can reduce the systems complexity at each point in time but of course it also adds additional complexity over time (e.g. if one allows for more than one reconfiguration at the same time within the system).

Having more than a few possible reconfigurations or reconfiguration needs to take place within small time segments, the reconfiguration to meet a new dynamic requirement cannot be done manually or interactively by the originator (e.g. the terminal user). In the next section the reconfiguration controller will be described.

5.5.7.1 Reconfiguration Controller

In our scenario three instances can initiate a reconfiguration:

- [1]. Client,
- [2]. Service or
- [3]. Observer.

To reconfigure the system depending on profile settings, an observer is needed which monitors the system environment and the communication profiles. The observer is a rule-based decision unit that can initiate a reconfiguration by sending a request to the reconfiguration controller.

In addition to this the client or a service can start a reconfiguration process. Figure 5-25 shows the three-reconfiguration possibilities.

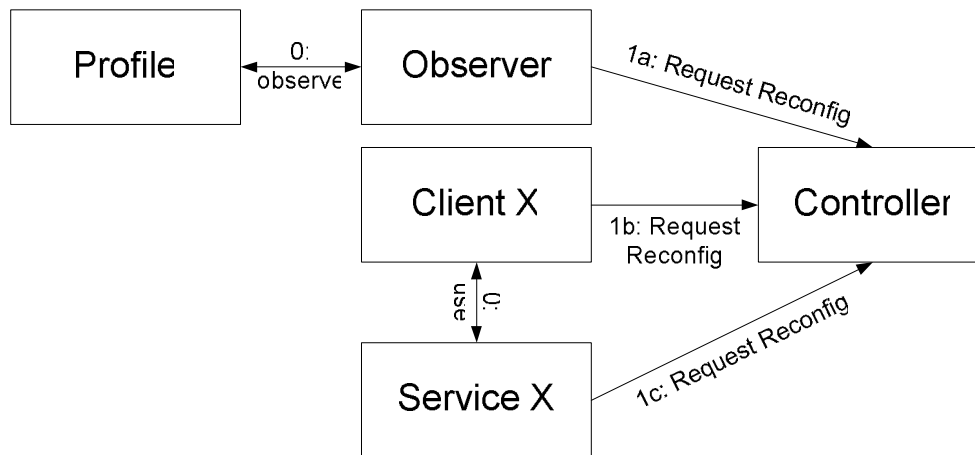


Figure 5-25: Reconfiguration Controller

In general we distinguish between system adaptation and application adaptation. System adaptation means the controller force a switching form one service to another one. A switching from HTML browsing with pictures (HTML1) to only HTML text browsing (HTML2) when the bandwidth is decreased will be demonstrated. Replacing the HTML1 service through the HTML2 service does this.

On the other hand the application adaptation does not force a service switching. Rather the application will be adapted to the new situation of the environment.

The system adaptation is the main focus; optionally application adaptation (DoCoMo contribution) will be implemented as described in 5.5.6.

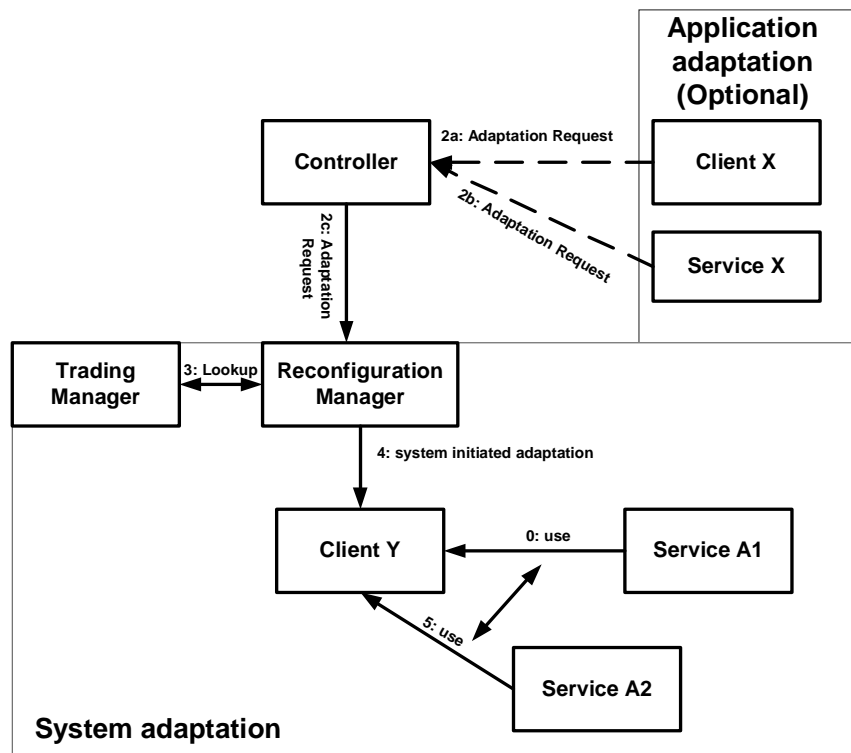


Figure 5-26: System Adaptation

5.5.8 Software Download

After the user initiated a download either directly by clicking a download link in the web browsing service or indirectly by selecting another service for usage that needs to download data to the user's terminal client (e.g. a GUI download), further interaction should not take place until the download has finished. We will software download by using our web browsing service. Through continuously browsing software downloading can be simulated. For our purpose this should be enough because we are more interested in measurements of delays and throughput than on a working download service.

5.5.9 Profiles

To realize an optimal service negotiation process the different "actors" of a mobile system that are described by communication profiles should be taken into considerations. A communication profile is an abstraction of all relevant participants in a mobile scenario, as there are the device, its network environment, the user and the available services on different levels ranging from lower layer bearer services to higher level teleservices and application services, the profiles contain all relevant information that will be needed for choosing a certain service. For the middleware demonstration, it is important that the user and application profiles should be implemented.

Profiles are modeled as services. The data can still be stored in a database. The service offers a generic API to access profile information using a database as backend for persistent data storage. The data-exchange format between the profile service and any clients is defined using XML. This is reasonable because the profile data structure represents a meta model that can hold any kind of information, for example about the user. XML is a suitable and expandable data format that can hold both content and definition of structure for the type of information to be exchanged. This is especially important to model a generic interface for accessing any kind of profile information. This is the same reason XML is nowadays used to exchange data in internet B2B applications. Instead of defining an API to access profile information for each type of profile and information type separately. The XML based approach allows for defining profile access based on a small number of basic interfaces.

This generic interface allows services that are a priori unknown to each other to communicate and decide about the type of profile information that can be understood by both parties at runtime

```
<?xml version="1.0" encoding="utf-8" ?>
<context>
  <namespace1:Element1 id="12345" type="entity">
```

```

        <namespace2.Element2
type="bool" >true</namespace2.Element2>
        <namespace3.Element3 type="ref" ></namespace3.Element3>
    </namespace1.Element1>
    <namespace3.Element3 type="int" >4</namespace3.Element3>
</context>

```

Figure 5-27: Example for an XML profile element

This is an example how a suitable definition of an XML profile can look like. The meaning of a class of elements is defined with the name space and identifier. This at the same time is a valid URI for a document that describe type and structure of the data contained in this element as well as the meaning if the information. This can be comparable to the specification of a service (interface and functionality)

The exact meaning of a profile element can be identified by its name and own namespace including namespaces of all parent nodes defined by the position of the profile element within the xml document which is in fact the representation of a tree. For an efficient handling, information about basic types can be included as xml attributes.

5.5.9.1 Application and User profile

The application profile is determined by the different application behavior towards the different QoS (in the demonstrator QoS refer to dynamic delay which is consisting of static delay and available user bandwidth) offered by the network. The application will adapt according to the QoS, an example for that would when downloading a HTML file that contains pictures. In the case that the network had offered a high data rate, i.e. less delay, the application may decide to download the complete file. On the other hand if the offered bandwidth is not sufficient to download the whole file and the delay will increase accordingly the application may decided to adapt to this situation by either pre-process the picture (down sampling) or leave them completely out and download only the text.

Different possibilities for the application behavior are:

- 1st possibility: Regular behavior that is independent from current QoS.
- 2nd possibility: Adaptive behavior at high (>384kbit/s), medium (64 – 384 kbit/s) and low bandwidth (10-64 kbit/s).

The Application behavior at different bandwidths will be:

- At high BW: the application behaves as in 1st possibility described
- At medium BW: the application will reduce picture size (sub-sampling and reduced resolution)
- At low bandwidth: the application will replace original pictures by small icons

To summarize the user profile and the application profiles are the triggers/inputs to the trading service. The important point for decision from the user point of view is the delay experienced. Delay is composed of two parts the static delay and the dynamic one. With static delay we understand the delay imposed by the server, we can we artificially delay the web browsing to emulate the core network (CN) delays caused by routers and other network elements. Possible values of the static delay are:

- NO delay,
- 1,
- 2 and
- 3 seconds.

The dynamic delay is defined over the download traffic and currently used bandwidth. Therefore the following BWs are required:

- 10,
- 56,
- 144,
- 384,
- 1000 and
- 11000 kbits/s.

The effective or composite delay is the sum of static and dynamic delay. The application on the client can only measure the composite delay, whereas the network part is able to differentiate the static and used BW, which result in a dynamic delay.

The following tables give an overview of the different user profile and application profile.

Table 5-4: User Profile

	Date Rate/Bandwidth	Delay	Priority
Business man profile	Demands the maximum	Less delay as possible	1

	rate can be offered	(preferable only the network latency)	
Medium class profile	Restricted to the maximum rate defined in the preferences (Billing reasons)	Can tolerate delay	2
Cheap class profile (student)	Can accept any rate offered by the network and his priority to negotiate is minimum	Can wait forever	3

Table 5-5: Application Profile

Cases	QoS offered	Action taken by the application
Case 1	High data rate, less delay	Successful download in reasonable time
Case 2	Medium data rate/tolerable delay	Adapting the application to the given QoS
Case 3	Low data rate/high delay	The application in this case is not taken any action it is up to the user to accept this service or to rejected.

In summary the communication profiles contain all relevant information that will be needed for choosing a certain service for an interaction or reconfiguration during a service interaction. Examples for profile information are capabilities of the current terminal, user preferences and current network bandwidth.

The decision process described in the requirement above ((Re-) configuration Decision) needs to have some kind of input data about the current system state or state of the system environment in the form of information about the system or environment entities that is stored in form of a profile based context model.

Since this is a demonstrator that shows scenarios, which are based on a middleware concept and not yet fully implemented, it is sufficient to prove that the implementation only works for the demonstration scenario. The implementation does not have to work for other possible scenarios.

5.5.9.2 Measuring Network QoS

For the demonstrator there are only two parameters that describe the network QoS. Bandwidth and delay. Both parameters are measured on the application layer by embedding a remote script into the transferred content that signals the server as soon as the page was completely loaded into the browser. The delay can be measured the same way by sending an (almost) empty page.

This mechanism has the advantage of working on most terminals with at least html3.2 capability (i.e. PocketPCs, Notebooks etc.) without installing or configuring additional software.

5.5.9.3 Measuring Application QoS

Measuring application level QoS is a quite complex task because it requires at least modification of the client application on the terminal. Since the scenarios are based on html this would mean modifying or implementing a new browser application. This is way out of scope for the resources available for the demonstrator development.

The mechanism described in 5.5.9.2 however allows for limited measuring of application QoS since it is based on the application layer (the http/html protocol). In combination with server side logic it is possible to distinguish between the text of a (html- not screen) page loaded and all/a number of pictures. The application QoS is therefore based on the time it takes to load the text of a page and load it completely. This way there are two parameters that can be used for reconfiguration decisions, e.g. switch to higher bandwidth if loading the text of a pages takes longer than two seconds, or strip away the pictures of a pages if loading the complete page takes longer than four seconds and there is no higher bandwidth available.

5.5.10 Logical Service System

After describing all components of the system we will give an overview of the architecture. In Figure 5-28 a detailed plan of the demonstrator is shown. The whole system contains of four units. The different communication profiles are blue colour. The reconfiguration controller and the trading service are orange. The application and the services are white and the simulation tool is light blue. The simulator is connected between the main communication channels to realise a flexible bandwidth adaptation.

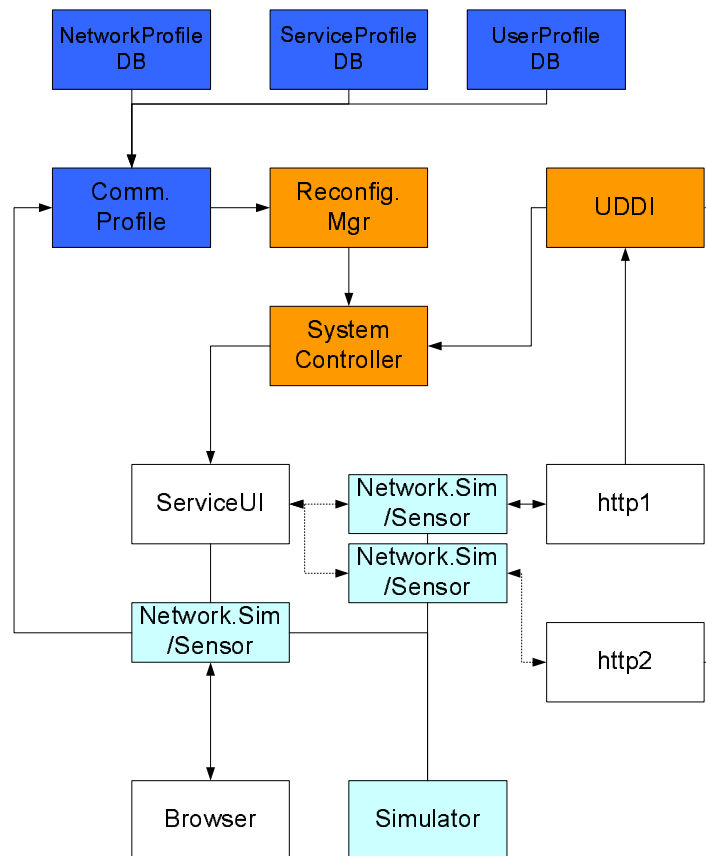


Figure 5-28: Logic Demonstrator

5.6 Evaluation Specification

Regarding the Evaluation Use Cases, the evaluation functionality of the demonstrator covers two major groups of parameters that need to be measured and the results displayed to the evaluation user of the demonstrator system. The first group is performance parameters that describe the timing behavior of the middleware demonstrator. These contain timing values for the overall reconfiguration process as well as timings for all three major sub components of the reconfiguration process, i.e. the access and distribution of context information in the communication profile on the one hand side and the trading or decision making process about a new system configuration and the process of changing an existing system to the new configuration on the other hand side.

As a short overview the following parameters are to be measured and will be displayed to the user:

- Time to detect reconfiguration necessity
- Time for reconfiguration/adaptation decision
- Time for reconfiguration deployment, or to adapt
- Maximum possible reconfiguration frequency.
- Average time for accessing current profile value
- Average time for accessing of history profile value

The second group of values to be measured contains values that affect scalability of the middleware. The first subgroup measures values that describe the amount of resources that are necessary to handle one user. The second subgroup measures the dependency between the most important performance values and the number of concurrent users in a system. In short the following scalability values are to be measured:

- Average memory usage for storing one profile data element
- Average size & memory usage for profile per user
- Average size & memory usage for profile history per user per day
- Average size of network profile
- Average size of terminal profile
- Average size of service profile
- Dependency between each timing value and number of concurrent users (optional).

Since all evaluation information will be measured with a demonstrator for reconfigurable middleware and because this demonstrator only emulates parts of a mobile telecommunication system infrastructure, the absolute numbers measured can not be transferred as absolute numbers to a real mobile telecommunications system infrastructure.

The measured values therefore should be seen as quality numbers that can show how certain parameters will scale, how effective certain optimisations compared to others and where the bottlenecks of the system and therefore candidates for further optimisations can be found.

5.6.1 General Reconfiguration Timing

This set of evaluation parameters is describing the process of system reconfiguration from a timing perspective. Typical sequences for reconfiguration already have been presented in 5.5.5.3. A typical reconfiguration that takes place during the current usage of a service by a client can be described regarding its timing behaviour as follows:

During the whole reconfiguration process there are three major important time points:

When a profile change reached the decision-making unit that decides about a reconfiguration. Usually the origin where a change in the context of a system is detected, i.e. the sensor for a specific kind of profile information, can be distributed all over the system. This means there is not only a delay between a context information changed and this change detected by the sensor, but also until it reaches the location it is processed.

When the decision-making unit has finished deciding about a new configuration. This of course depends on several things like the complexity of the decision itself and any further profile information that might be involved in this decision.

When the system has been completely changed according to the new configuration and any suspended session was successfully continued if possible.

With this three-time point during a reconfiguration it is possible to define two different kinds of reconfiguration times. The first one in chase the system situation changed in a way that it is still possible to maintain all current services bound to any client, but with maybe lower QoS (Shadow Reconfiguration). The second case is that the system situation changed in a way that makes it impossible to continue the current service sessions until the system is reconfigured (Hot Reconfiguration). The actual Handover Time is defined as the time of interruption of current service usage's due to any kind of reconfiguration. For cases of hot reconfiguration the handover time is equivalent to the hot reconfiguration time.

All time spans between the mentioned time points and reconfiguration- and handover times can be further broken up into timing evaluation parameters according to the next sections.

The most important parameters are described split into three groups, the timings related to detecting and signaling context information into and from the profiles, timings that describe the actual trading process and timings for the deployment of configuration changes.

5.6.2 Profile specific Timings

This section describes timings related to detecting and signalling context information into and from the profiles in more detail as well as how they are measured during the evaluation.

5.6.2.1 Measuring Profile Element Change Detection

Parameter Description:

This is in general the time necessary to detect a change of the situation the system is in. This can be both, internal or external state, i.e. the state of the environment of the system. The situation of a system is described as its context. Usually a sensor or a probe detects part of the systems context.

Measurement Specification:

The *Profile Element Change Detection Time* is the time between context information changes in the physical reality and a sensor can detect this change. This of course depends on the type of sensor and type of context information that should be detected.

For this demonstrator the two-network parameters delay and bandwidth can be simulated. *Profile Element Change Detection Time* will be measured after sending the simulation command until the sensor reports a change detected. The communication delay between measuring, simulation and sensor component will be subtracted after measuring an empty roundtrip between these three components.

5.6.2.2 Measuring Profile Element Change Propagation

Parameter Description:

After a sensor or a probe has detected part of the systems context, this information is stored in a profile that describes one entity within the systems context. In case of this demonstrator there are four types of

profiles. Network, service, user and terminal profiles. These profiles it selves and also each single profile can be distributed within the system, for example the static parts of the user profile can be stored in his home network and his dynamic profile on the terminal he is currently using.

Measurement Specification:

The Profile Element Change Propagation Time is the time between a sensor detecting a change and propagating it to the unit that decides whether a reconfiguration is necessary or not. Since there can be a chain of units in between that interpret profile information produced by sensors into higher level profile information, like for example reasoning about changes of user preferences depending on certain profile information, this parameter is not easy to evaluate. For this demonstrator the time is measured between the two sensors for network bandwidth and delay and the first and last unit the uses this profile information. The last unit is always the decision unit within the Reconfiguration Manager that decides about a reconfiguration.

5.6.2.3 Measuring Profile Element Access

Parameter Description:

Since the actual values of certain profile elements can be distributed in the system, there is a time span between requesting the current value of a profile element.

During the process of deciding about a new configuration the Reconfiguration Manager that was first invoked by a change of another profile element might need additional profile information to decide about a change of configuration. This information can be cached locally in the profile access service or requested just in time through the Active Attribute Mechanism, depending on the type of context information.

Measurement Specification:

The Profile Element Access Time is the time between requesting the value of a profile element and receiving the result. In the demonstrator this parameter is measured for each stored parameter in one profile for each type of profiles during different states of the system, for example during idle time, during application usage and during reconfiguration.

Post processing:

Results are grouped according to the type of Active Attribute (cached, predicted, just-in-time etc.) that was used to access the profile element. Results are presented as a table.

5.6.3 Trading specific Timings

This section describes timings that describe the actual trading process in more detail as well as how they are measured during the evaluation.

5.6.3.1 Measuring Reconfiguration Computation

Parameter Description:

After the decision process about a reconfiguration was initiated either by a change of a profile value or on a periodical basis there is a time span that describes a set of calculations that produce a new logical (i.e. defined in terms of functional requirements in form of service bindings and optional non-functional requirements that will later restrict mapping to existing components in the trading phase) system configuration.

Measurement Specification:

The Reconfiguration Computation Time is the time between a physical change of the system situation until a new configuration is decided but not yet mapped to components and deployed minus the time for detecting and propagating the changed profile information and minus any necessary access of additional profile information.

To ease measuring in the demonstrator all reconfiguration decisions in the evaluation scenarios will be implemented so that any additional profile information is accessed in advance before the actual computation of the new configuration starts. Measurement stops when the algorithm terminates.

There are three different browsing services (full pictures, compressed pictures, without pictures). This will translate to six mandatory reconfiguration scenarios. That can be activated either by a change of bandwidth or delay. Time for each reconfiguration decision is recorded separately.

5.6.3.2 Measuring Trading Computation

Parameter Description:

After the Reconfiguration Manager decided about a new logical service configuration, this configuration needs to be mapped to actual components that offer the requested functionality and non-functional requirements. This results in several binding commands between components in the roles of service clients and services.

Measurement Specification:

The Trading Computation Time is the time necessary for mapping services onto components and resolving any further functional dependencies or interactions that might result from that mapping. It is measured between the Reconfiguration Manager finished its decision about a new configuration and the trading manager generating the mapping and resolving the dependencies, but before the binding commands are actually sent out to the different components.

Again this is done for each of the six mandatory reconfiguration scenarios of the demonstrator.

5.6.3.3 Measuring Trading Deployment**Parameter Description:**

After the new configuration was translated into a set of binding commands these commands need to be transferred to all involved components and these components need to carry out this commands. There might be additional steps involved to establish a transaction mechanism and handover of session states.

Measurement Specification:

The Trading Deployment Time is the time between transferring all binding commands and receiving confirmation that they have been applied to the components. For the demonstrator the transaction and session transfers are optional. The parameter is recorded for each of the six mandatory reconfiguration scenarios.

5.6.4 Scalability Evaluation

This section describes parameters that describe on the one hand necessary resources to store profile information on a per user and per service basis as well as dependencies between previously described timing parameters and number of concurrent users and available services.

5.6.4.1 Measuring Profile Storage Resources**Parameter Description:**

Each profile element requires a certain amount of memory. Each user and each available service has its own profile therefore the number of users and available services can greatly affect the scalability of a system

Measurement Specification:

For the demonstrator the necessary storage for each profile can be calculated from the actual implementation.

5.6.4.2 Measuring User dependency**Parameter Description:**

The number of users in a system can greatly affect overall reconfiguration performance either because additional profile parameters need to be taken account into the reconfiguration decision or because of the overall number of concurrent reconfigurations.

Measurement Specification:

The user dependency is first how many concurrent reconfigurations take place in average, second how this affects the performance of on specific reconfiguration. In a good case this dependency scales linear. Each timing parameter during a reconfiguration process is measured with a varying number of concurrent users/reconfigurations.

Post processing:

Dependencies are shown as a table and optionally as a chart.

5.7 Middleware Platform**5.7.1 Hardware Deployment**

In Figure 5-29 the most likely hardware structure of the demonstrator is shown. There is one server on the fixed network that represents the home network and holds the HRM. A laptop server that holds a PRM and is associated to an 802.11 wireless network access point represents each location. The PRM laptops can be connected to the HRM over an Intranet or the Internet. This way the demonstrator can easily be presented in different physical locations.

The mobile terminals can roam between the two PRM networks. The PRM network is also used to determine the location of a terminal.

However since the demonstrator itself is modular and scaleable, more than two PRM networks can be added by adding either additional Access Points and Laptop servers or only APs (the laptop server then holds more than one instance of a PRM implementation).

For minimal demonstration purposes it is also possible to run instances of HRM and two PRMs on one laptop server.

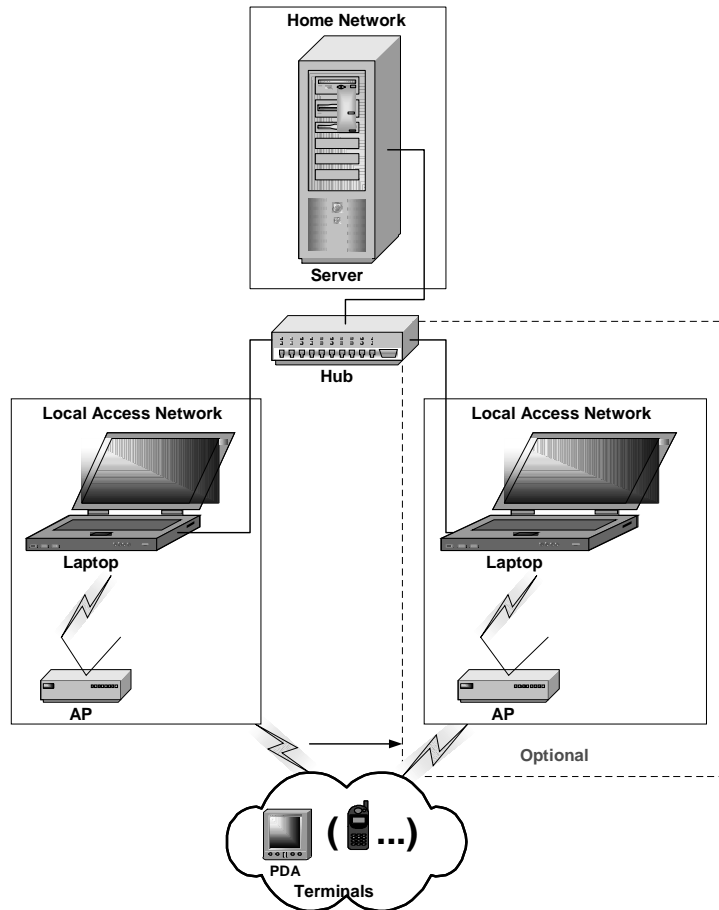


Figure 5-29: Hardware Structure of Demonstrator

5.7.2 Microsoft .Net, SOAP and Web services

The demonstrator will be implemented using the Microsoft .Net platform since it offers the necessary middleware functionality (web services, remoting, xml) and is significantly faster and easier to apply than the alternative J2EE platform³. However there will be a demonstrator API to allow for easy extension of the demonstrator. This demonstrator API will rely on open standards for distributing functionality (Web services, SOAP and XML). This means all parts of the demonstrator that export their functionality and interfaces for the demonstrator API use a XML-RPC based mechanism (SOAP) that is based on the http protocol to accept remote procedure calls from other distributed components. All components in the API are described as webservices using the WSDL standard for describing their exported interface and functionality³.

The demonstrator itself is bound to a Microsoft Windows platform so far (until the .Net runtime for Linux is finished), but because the platform independent RPC mechanism all major components can also be accessed by Java-based and therefore platform independent implementations. This is not necessary for the scenarios described in this document but could be used for future extensions or other SCOUT work-packages that would like to integrate their demonstrations within the evaluation and presentation scenarios specified in this document.

5.8 Conclusion

In this chapter the general objectives to adequately demonstrate a middleware concept for reconfiguration have been described. Three groups of objectives were described:

- Visualizations:
- Validation.

³ This is comparable to an IDL

- Evaluation:

Based on these objectives, a set of application scenarios was described to support visualization objectives as well as evaluation of the overall middleware concept based on measuring and post processing statistical data that give an idea about the performance and scalability of using the middleware technology in wireless environment.

Additionally a set of key functional requirements was presented, that describe key functions of the middleware, the user interface and the instrumentation to gather evaluation data.

In a next step, a specification of the demonstrator is developed based on the scenarios and functional key requirements. The main system components: **Applications, Terminal Management Simulation, Trading, Reconfiguration, Evaluation and Profiles** have been specified. These specifications will serve as basis for the implementation phase.

5.9 References

- [5-1] IST-1999-12070 TRUST, D4.3 "Report on Assessments of Novel Solutions on System Aspects of Reconfigurable Terminals and Recommendation for standardization", WP4, Oct 2001.
- [5-2] IST-2001-34091 SCOUT, D4.1.1 "Requirements on Network and Security Architecture and Traffic Management Schemes for Download Traffic based on IP Principles in Cellular and Ad Hoc Networks", WP4, Oct 2002.
- [5-3] www.uddi.org
- [5-4] 3GPP TS 23.057 v4.5.0 "Mobile Execution Environment (MExE); Functional description", March 2002.
- [5-5] E. Mohyeldin, M. Fahrmaier, P. Donrbush, M. Dillinger, J. Luo and C. Salzmann "Communication Profiles for SDR Equipment", IST Summit 2002.
- [5-6] Cawar. <http://www.cawar.de>.
- [5-7] Micheal Fahrmaier, Chris Salzmann, Maurice Schoenmakers. "Carp@ - Managing Dynamic Jini Systems" Proceedings of Middleware 2000 - Work in Progress Track. New York, NY Apr. 2000
- [5-8] Micheal Fahrmaier, Chris Salzmann, Maurice Schoenmakers. "A Reflection Based Tool for Observing Jini Services. In Cazzola et al. "Reflection and Software Engineering.LNCS 1826, Springer Verlag June 2000.
- [5-9] Max Breitling, Michael Fahrmaier, Chris Salzmann, Maurice. "Carp@ - Managing Dynamic Distributed Jini Systems". Proceedings of OORaSE'99 - OOPSLA Workshop on Object Oriented Reflection and Software Engineering, Denver, CO November 1999.
- [5-10] <http://www.middleware-company.com/j2eedotnetbench/>.

6. Conclusions

The expected results of these various demonstrations are:

1/ Reconfigurable RF TX chain

The demonstrator model, according to the previously presented analysis, can prove all the significant parameters of a reconfigurable terminal (channel shadowing, output power, phase noise, linearity, etc.). The performance can be demonstrated at three different standards (GSM, UMTS and Hyperlan2). The model shall be realised with a general purpose (not a particular design is needed): the H/W shall satisfy all the standards requirements.

2/ Link adaptation on hardware validator

The experimentation that has been carried out during TRUST project has shown that the adaptive modulation supplies a high gain onto the link budget for radio link on multipath propagation channels with Doppler. When using adaptive modulation and adaptive error correction coding, it is expected to obtain a higher gain. The experimentation to be led in SCOUT project will allow to determine the conditions of a realistic use of the link adaptation. Moreover the interaction of the link adaptation with the scalable video transmission will be studied in order to see what sort of improvement can be provided.

3/ Video demonstration

The SCOUT video demonstration over the hardware validator requires integration of the video codecs (UoB A4.3.4 and 4.3.5) and the hardware validator (FT R&D A4.3.3). The aim is to provide a simple demonstration system which can effectively demonstrate the technology considered in A4.3.3, A4.3.4 and A4.3.5 without the need for a complicated interface or implementation of a complete protocol stack.

4/ Middleware demonstrator

The general objectives to adequately demonstrate a middleware concept for reconfiguration have been described. Three groups of objectives were described:

- Visualizations:
- Validation.
- Evaluation:

Based on these objectives, a set of application scenarios was described to support visualization objectives as well as evaluation of the overall middleware concept based on measuring and post processing statistical data that give an idea about the performance and scalability of using the middleware technology in wireless environment.

Additionally a set of key functional requirements was presented, that describe key functions of the middleware, the user interface and the instrumentation to gather evaluation data.

In a next step, a specification of the demonstrator is developed based on the scenarios and functional key requirements. The main system components: **Applications, Terminal Management Simulation, Trading, Reconfiguration, Evaluation and Profiles** have been specified. These specifications will serve as basis for the implementation phase.