



OverDRiVE

Spectrum Efficient Uni- and Multicast Services
over Dynamic Multi-Radio Networks in Vehicular Environments

Description of Demonstrator for D14 Mobile Multicast and the Vehicular Router



11 0001 0 010 100 010 00 0001 01
01 0 010 100 010 00



IST-2001-35125 (OverDRiVE)

D14

Description of Demonstrator for Mobile Multicast and the Vehicular Router

| | |
|---|--|
| Contractual Date of Delivery to the CEC: | 02/2004 (Project month #23) |
| Actual Date of Delivery to the CEC | 02/2004 |
| Author(s): | Tim Leinmüller (Editor, DC) |
| Participant(s): | Ericsson, DaimlerChrysler, Motorola, Rai, University of Bonn |
| Project Title: | Spectrum Efficient Uni- and Multicast Services over Dynamic Multi-Radio Networks in Vehicular Environments |
| Workpackage contributing to the Document: | WP2/WP3 |
| Estimated Person Month: | 88 (48 WP2 / 40 WP3) |
| Security Type: (Int/Res/IST/FP5/Pub)¹ | Pub |
| Document Number²: | IST-2001-35125/OverDRiVE/WP2-3/D14 |
| Nature of the Document³: | (R)eport |
| Version (Status of the Document: D1/R1/D2/R2/F)⁴: | F |
| Total number of pages: | 114 |

¹ *Int* Internal circulation within project (and Commission project Officer if requested)

Res Restricted circulation list (specify in footnote) and Commission PO only

IST Circulation within IST Programme participants

FP5 Circulation within Framework Programme participants

Pub Public document

² Format: IST-2001-35125/OverDRiVE/<source>/<Deliverable Number: Dxx | Running Number for other: Rxx>:

Example: IST-2001-35125/OverDRiVE /WP1/D01 (Document comes from the WP1)

³ (R)eport, (P)rototype, (D)emonstrator, (S)pecification, (T)ool, (O)ther

⁴ V0.x=Draft, V1.x=Final. (D1=First Draft, R1=Technically Revised, D2=Final Draft, R2=Final Revised, F=Final)

Abstract:

The deliverable describes the demonstration and verification efforts of the IST 5th Framework Programme project OverDRiVE (Spectrum Efficient Uni- and Multicast Services Over Dynamic Radio Networks in Vehicular Environments). The demonstration activities were focused on mobile multicast and mobile router/networks workpackages. The demonstrator was developed in evolutionary steps towards the final demonstration as it was demonstrated at the HyWiN 2003 workshop and the annual project review in Torino, December 2003. The demonstrator helped to verify key aspects of the OverDRiVE concepts and aims at raising the interest of the scientific community with respect to the project results.

Keyword List: Mobile Multimedia, Mobile Multicast, Mobile Networks, Mobile Router, Mobile IPv6, Group Management, Handover, WLAN, UMTS, DVB/T

Authors

Tim Leinmüller (Editor, DaimlerChrysler AG)
Michael Wolf (DaimlerChrysler AG)
Christian Maihöfer (DaimlerChrysler AG)
Christoph Barz (University of Bonn)
Markus Pilz (University of Bonn)
Christophe Janneteau (Motorola Labs, Paris)
Alexandru Petrescu (Motorola Labs, Paris)
Paolo Casagrande (Radiotelevisione Italiana, Rai)
Andrea Bertella (Radiotelevisione Italiana, Rai)
Miklós Aurél Rónai (Ericsson Research Hungary, Traffic Lab)
Kristóf Fodor (Ericsson Research Hungary, Traffic Lab)
Gergely Biczók (Ericsson Research Hungary, Traffic Lab)
Csaba Simon (Budapest University of Technology and Economics)
Péter Kersch (Budapest University of Technology and Economics)
Rolland Vida (Budapest University of Technology and Economics)
Ralf Tönjes (Ericsson Eurolab Germany)
Thorsten Lohmar (Ericsson Eurolab Germany)

Revision History

| Revision | Date | Issued by | Description |
|------------|------------|--------------------------------------|--|
| V0.01 (D1) | 2003-04-08 | Christian Maihöfer | Started Document |
| | 2004-01-14 | Christophe Janneteau | Section 5.2: Mobility of IPv6 Multicast Communications. |
| | 2004-01-15 | Paolo Casagrande | Section 8.1, Section 8.2, some new references. |
| | 2004-01-20 | Tim Leinmüller | Annex D, Section 4.4, 4.5. |
| | 2004-01-21 | Miklós Aurél Rónai | Added ETH authors and section 3 (pre-audit demos). |
| | 2004-01-23 | Paolo Casagrande, Andrea Bertella | Section 7.2. DVB-T part completed. Section 8.2.5: Detailed DVB-T setup used in Turin. Annex F: BTFTP specification. |
| V0.05 | 2004-02-03 | Miklós Aurél Rónai | Rewritten ETH part merged. |
| | 2004-02-05 | Tim Leinmüller | Editorial changes. |
| | 2004-02-11 | Rolland Vida | Sections 5.3 and 8.3: Modified BUTE's, moved simulation results to D16. |
| | 2004-02-17 | Paolo Casagrande | Section 8.1.1: 6Bone. Section 7.3: Web Broadcast. Reviewed section 7.2. |
| | 2004-02-18 | Alex Petrescu | Section 5.1: Mobility of IPv6 Unicast Communications. Section 3.3.2: CRM demo at IST Mobile Summit and IST Summit. Inserted partner responsibility for UMTS and GPRS FrontBoxes for CRM and RAI. |
| V0.06 | 2004-02-19 | Tim Leinmüller | Merged different version for a first quite stable version. |
| V0.07 | 2004-02-25 | Michael Wolf, Tim Leinmüller | Final editorial changes. |
| V0.08 | 2004-03-03 | Tim Leinmüller | Fixed file format error. |
| | 2004-03-04 | Alex Petrescu | Review for contents, grammar and uniformity. |
| | 2004-03-04 | Miklós Aurél Rónai | Made some editorial changes. |
| V0.09 (R1) | 2004-03-15 | Tim Leinmüller | Changes resulting from review. |
| V1.00 (F) | 2004-03-19 | Tim Leinmüller | Final Version. |

Table of Contents

| | |
|--|-----------|
| Abstract | 2 |
| Authors | 3 |
| Table of Figures | 9 |
| Acronyms | 11 |
| Executive Summary | 14 |
| 1 Introduction | 15 |
| 2 Mobile Router Scenario and Demo Story | 16 |
| 2.1 Technical Description | 17 |
| 3 Demonstrator Development Description | 19 |
| 3.1 Introduction | 19 |
| 3.2 Theoretical Overview of Ericsson Traffic Lab’s OverDRiVE Moving Network Testbed | 19 |
| 3.3 IST Mobile and Wireless Communications Summit 2003 | 20 |
| 3.3.1 Moving Network Testbed of Ericsson Hungary at the Mobile Summit 2003..... | 20 |
| 3.3.2 CRM Demo..... | 22 |
| 3.4 CRM Field Trials in Paris | 23 |
| 3.4.1 Moving Network..... | 24 |
| 3.4.2 Home Network..... | 26 |
| 3.4.3 GPRS Network..... | 27 |
| 3.4.4 WLAN HotSpot Network..... | 28 |
| 3.4.5 UDP Tunnel Software..... | 29 |
| 3.4.6 Mobility Scenario..... | 29 |
| 3.5 OverDRiVE Project Meeting Budapest | 31 |
| 3.6 PCC Wireless Communications Research Days | 33 |
| 4 Common Demonstrator Architecture | 36 |
| 4.1 Demonstrator Setup | 36 |
| 5 Mobility Management | 39 |
| 5.1 Mobility of IPv6 Unicast Communications | 39 |
| 5.1.1 LIVSIX IPv6 Stack..... | 39 |
| 5.1.2 IPv6 Router..... | 40 |
| 5.1.3 Routing Advertisement Module..... | 40 |
| 5.1.4 Routing Module..... | 41 |
| 5.1.5 Mobile Router (at home)..... | 42 |
| 5.1.6 Mobile Router (in a foreign network)..... | 42 |
| 5.1.7 Home Agent for a Mobile Router..... | 43 |
| 5.2 Mobility of IPv6 Multicast Communications | 44 |
| 5.2.1 Demonstration Story..... | 44 |
| 5.2.2 Demonstrator Configuration..... | 46 |
| 5.2.3 HyWIN 2003 Demonstrator..... | 48 |

| | | |
|------------|---|-----------|
| 5.3 | Seamless IPv6 Multicast Handovers | 54 |
| 5.3.1 | Demonstration | 55 |
| 5.3.2 | Handovers..... | 57 |
| 6 | <i>Group Management for Mobile Multicast</i> | 58 |
| 7 | <i>Demonstrator Services</i> | 61 |
| 7.1 | Introduction | 61 |
| 7.2 | Video Streaming | 61 |
| 7.2.1 | Realization | 61 |
| 7.2.2 | IVAN Set Up | 62 |
| 7.2.3 | Content Server | 63 |
| 7.2.4 | Content | 63 |
| 7.3 | Web Broadcast over a DVB-T cell | 64 |
| 7.3.1 | Realization | 64 |
| 7.3.2 | IVAN Set Up | 64 |
| 7.3.3 | Content Server | 65 |
| 7.4 | Multicast Messaging and Streaming | 66 |
| 7.4.1 | Realization | 66 |
| 7.4.2 | Content Server | 67 |
| 7.4.3 | Content | 67 |
| 7.5 | Adaptive Video on Demand | 67 |
| 7.5.1 | Realization | 67 |
| 7.5.2 | IVAN Set Up | 68 |
| 7.5.3 | Content Server | 68 |
| 7.5.4 | Content | 68 |
| 7.6 | Remote Access | 69 |
| 7.6.1 | Realization | 69 |
| 7.6.2 | IVAN Set Up | 70 |
| 7.6.3 | Content Server | 70 |
| 7.6.4 | Content | 70 |
| 7.7 | Software Download | 72 |
| 7.7.1 | Realization | 72 |
| 7.7.2 | IVAN Set Up | 72 |
| 7.7.3 | Content Server | 72 |
| 7.7.4 | Content | 73 |
| 7.8 | Web Access | 73 |
| 7.8.1 | Realization | 73 |
| 7.8.2 | IVAN Set Up | 74 |
| 7.8.3 | Content Server | 75 |
| 7.8.4 | Content | 75 |
| 8 | <i>External Interfaces</i> | 76 |
| 8.1 | Network Architecture | 76 |
| 8.1.1 | 6Bone, architecture figure and description | 76 |
| 8.1.2 | IP/DVB Gateway | 77 |
| 8.2 | DVB | 77 |
| 8.2.1 | DVB Front box | 77 |
| 8.2.2 | Forwarding the received IPv6 packets: fwdsix | 79 |
| 8.2.3 | Architecture description..... | 79 |

| | | |
|----------------|---|------------|
| 8.2.4 | IPv6 over IPv4 Encapsulator Setup: mproxy | 79 |
| 8.2.5 | DVB Platform | 80 |
| 8.3 | GPRS and UMTS | 82 |
| 8.3.1 | GPRS | 82 |
| 8.3.2 | UMTS | 86 |
| 8.4 | WLAN..... | 86 |
| 9 | <i>Internal Interfaces.....</i> | 87 |
| 9.1 | IVAN Architecture | 87 |
| 9.2 | CAN Bus interface..... | 87 |
| 9.3 | Bluetooth | 88 |
| 9.3.1 | Bluetooth Hardware for BlueZ | 88 |
| 9.3.2 | Creating a software bridge | 88 |
| 9.3.3 | Prepare for upcoming PAN connections..... | 89 |
| 9.3.4 | Enabling and connecting to a NAP | 89 |
| 9.4 | WLAN..... | 89 |
| 10 | <i>Conclusion.....</i> | 90 |
| | <i>References</i> | 91 |
| Annex A | <i>Software Development for Linux-based Handhelds.....</i> | 93 |
| A.1 | Introduction | 93 |
| A.2 | Cross-Compiling..... | 93 |
| A.2.1 | ipkg-utils and ipkg..... | 93 |
| A.3 | Cross-Compiling the Linux kernel..... | 94 |
| A.4 | Create Kernel Packages | 94 |
| A.5 | Compile Sound Module..... | 94 |
| A.6 | Images for the iPAQ..... | 95 |
| A.6.1 | Mounting an jffs2 image | 95 |
| A.6.2 | Modify an iPAQ image | 95 |
| A.6.3 | Booting the Image | 96 |
| A.7 | Native compiling | 96 |
| A.8 | Example: Cross-Compiling Livsix for the iPAQ | 96 |
| A.9 | Example: Compiling mpeg4ip on a intimate Linux iPAQ..... | 97 |
| Annex B | <i>Software Update - Setup and Configuration.....</i> | 98 |
| B.1 | Installation guide | 98 |
| B.1.1 | Java SDK installation | 98 |
| B.1.2 | Tomcat 4.1.24 installation..... | 98 |
| B.1.3 | ant installation | 100 |
| B.1.4 | Update server and Web-interface installation | 100 |
| B.2 | Configuration guide..... | 101 |
| B.2.1 | The UpdateServer configuration guide | 101 |
| B.2.2 | The Client-Web-Interface configuration guide | 102 |
| Annex C | <i>Broadcast Trivial File Transfer Protocol Specification.....</i> | 104 |

Annex D Responsibilities of Partners 114

Table of Figures

| | |
|--|----|
| Figure 1: PDAs inside their Home Network | 17 |
| Figure 2: Nested Tunnels..... | 18 |
| Figure 3: Theoretical overview of the MRHA – BCMP combined solution..... | 20 |
| Figure 4: Ericsson’s OverDRiVE testbed overview (IST Mobile Summit 2003)..... | 21 |
| Figure 5: Network topology of the Ericsson testbed (IST Mobile Summit 2003) | 22 |
| Figure 6: Overview of Field Experiments with a Moving Network..... | 24 |
| Figure 7: Moving Network..... | 25 |
| Figure 8: Envisioned Mobile Router | 26 |
| Figure 9: Home Network..... | 26 |
| Figure 10: GPRS Network..... | 27 |
| Figure 11: WLAN HotSpot Network | 28 |
| Figure 12: Mobility Scenario and Dynamics of IP Address Assignment..... | 29 |
| Figure 13: Tunnelling Dynamics during IP Mobility Scenario | 30 |
| Figure 14: Ericsson OverDRiVE testbed overview (Budapest meeting) | 31 |
| Figure 15: Network topology of the Ericsson testbed (Budapest meeting)..... | 33 |
| Figure 16: Ericsson’s OverDRiVE testbed overview (PCC Workshop)..... | 35 |
| Figure 17: Network topology of the Ericsson testbed (PCC Workshop) | 35 |
| Figure 18: Common Demonstrator Architecture..... | 36 |
| Figure 19: Demonstration Setup..... | 37 |
| Figure 20: Functional Components | 38 |
| Figure 21: IPv6 multicast for moving networks – Full demonstrator architecture..... | 46 |
| Figure 22: IPv6 multicast for moving networks – HyWIN 2003 demonstrator | 49 |
| Figure 23: MN and MR Multicast Handover GUIs – Startup. | 50 |
| Figure 24: MN and MR Multicast Handover GUIs – MN starts video streaming client. | 51 |
| Figure 25: MN and MR Multicast Handover GUIs – MN moves into the moving network..... | 51 |
| Figure 26: MN and MR Multicast Handover GUIs – MR moves to Outdoor WLAN #2..... | 52 |
| Figure 27: MN and MR Multicast Handover GUIs – MR moves to DVB-T..... | 52 |
| Figure 28: Poster of the “Multicast for Moving Network” demonstration at HyWIN 2003..... | 53 |
| Figure 29: Multicast Routers in the Access Network..... | 54 |
| Figure 30: Seamless Multicast Handover Solution | 55 |
| Figure 31: The topology of the testbed..... | 56 |
| Figure 32: The GUI for multicast handover control..... | 56 |
| Figure 33: Handover types for the Mobile Terminal..... | 57 |
| Figure 34: Group Management lab setup | 58 |
| Figure 35: Screen-Shot of the Group Management GUI..... | 59 |
| Figure 36: Topology of the Group Management Demonstration | 60 |
| Figure 37: DVB-T Video Screenshots | 62 |
| Figure 38: Basic architecture scheme used to broadcast Web pages over a DVB-T cell..... | 64 |
| Figure 39: Reception of broadcast Web Pages over DVB-T..... | 65 |
| Figure 40: Example Web Pages | 65 |
| Figure 41: Demonstration Control Application..... | 66 |
| Figure 42: Adaptive Video on Demand..... | 68 |
| Figure 43: Chan Chan videos - 3 different qualities..... | 69 |
| Figure 44: Remote Access - Travel Data..... | 71 |
| Figure 45: Remote Access - Convenience..... | 71 |
| Figure 46: Remote Update with Java XML Messaging | 72 |

| | |
|---|----|
| Figure 47: Remote Update..... | 73 |
| Figure 48: HTTP-proxy demo setup..... | 74 |
| Figure 49: Access to the 6Bone from the Rai Crit site, during the demo..... | 76 |
| Figure 50: DVB-T Front Box, receiver side..... | 78 |
| Figure 51: DVB-T Front Box, receiver side..... | 79 |
| Figure 52: DVB encoding and multiplexing chain..... | 80 |
| Figure 53: The GPRS FrontBox Architecture..... | 83 |
| Figure 54: The connections in the architecture..... | 84 |
| Figure 55: IVAN Architecture..... | 87 |

Acronyms

| | |
|--------|---|
| AAA | Authentication, Authorisation, Accounting |
| AAAF | Foreign AAA server |
| AAAH | Home AAA server |
| AAAL | Local AAA server |
| ACK | Acknowledgement |
| ACS | Access System |
| AH | Authentication Header |
| ANP | ANchor Point |
| APS | Application Service Provider |
| AR | Access Router |
| ARQ | Automatic Repeat reQuest |
| AS | Attachment Server |
| BACk | Binding Acknowledgement |
| BAN | Body Area Network |
| BCMP | BRAIN Candidate Mobility Protocol |
| BG | Border gateway |
| BN | Buffering Node |
| BR | Border Router |
| BRAIN | Broadband Radio Access for IP based Networks |
| BReq | Binding Request |
| BU | Binding Update |
| CA | Conditional Access |
| CAN | Controller Area Network |
| CAS | Conditional Access System |
| CN | Correspondent Node |
| CoA | Care-of-Address |
| CPU | Central Processing Unit |
| DHCP | Dynamic Host Configuration Protocol |
| DMA | Domain Multicast Agent |
| DNS | Domain Name Service |
| DR | Designated Router |
| DRiVE | Dynamic Radio for IP-Services in Vehicular Environments |
| DVB | Digital Video Broadcast |
| DVB-T | Terrestrial Digital Video Broadcasting |
| DVMRP | Distance Vector Multicast Routing protocol |
| EAP | Extensible Authentication Protocol |
| EAPOL | EAP Over LAN |
| ECM | Entitlement Control Message |
| EMM | Entitlement Management Messages |
| ESP | Encapsulating Security Protocol |
| FEC | Forward Error Correction |
| FMIP | Fast Mobile IP |
| FR | Foreign Router |
| GPRS | General Packet Radio Service |
| GW | Gateway |
| HA | Home Agent |
| HMIP | Hierarchical Mobile IP |
| HMIPv6 | Hierarchical Mobile IP version 6 |
| HO | Handover |

| | |
|-------|---|
| HoA | Home Address |
| HSS | Home Subscriber Server |
| ICMP | Internet Control Message Protocol |
| IETF | Internet Engineering Task Force |
| IGMP | Internet Group Management Protocol |
| IKE | Internet Key Exchange |
| IP | Internet Protocol |
| IPSEC | Internet Security Protocol |
| IPsec | Internet Protocol Security |
| IPv4 | IP version 4 |
| IPv6 | IP version 6 |
| ISP | Internet Service Provider |
| IST | Information Society Technologies |
| IVAN | Intra Vehicular Area Network |
| LAN | Local Area Network |
| LCoA | Local Care-of-Address |
| LFN | Local Fixed Node |
| LMA | Local Multicast Agent |
| LMN | Local Mobile Node |
| LMR | Local Multicast Router |
| LS | Logging Server |
| MA | Multicast Agent |
| MANET | Mobile Ad hoc NETWORKS |
| MAP | Mobility Anchor Point |
| MC | Multicast |
| MH | Mobile Host |
| MIND | Mobile IP-based Network Development |
| MIP | Mobile IP |
| MIPv6 | Mobile IP version 6 |
| MLD | Multicast Listener Discovery |
| MMP | Multicast for Mobility Protocol |
| MN | Mobile Node |
| MNN | Mobile Network Nodes |
| MOSPF | Multicast Extensions to OSPF |
| MOST | Media Oriented System Transport |
| MR | Mobile Router |
| MRHA | Mobile Router – Home Agent bidirectional tunnel |
| M RTP | Mobile Router Tunnelling Protocol |
| MS | Mobile Station |
| MSDP | Multicast Source Discovery Protocol |
| MSN | Multi Access Support Node |
| MT | Mobile Terminal |
| MTU | Maximum Transfer Unit |
| MU | Mobile User |
| NACK | Negative Acknowledgement |
| NAR | New Access Router |
| NAT | Native Address Translation |
| ND | Neighbour Discovery |
| NEMO | NETwork MOBility (working group) |
| NLRM | New Local Multicast Router |

| | |
|------------------|--|
| OverDRiVE | Spectrum Efficient Uni- and Multicast Services over Dynamic Multicast Services Over Dynamic Radio Networks in Vehicular Environments |
| OVR | OverDRiVE Resource |
| PAN | Personal Area Network |
| PANA | Protocol for carrying Authentication for Network Access |
| PAR | Previous Access Router |
| PDA | Personal Digital Assistant |
| PIM- $\{SM DM\}$ | Protocol Independent Multicast – {Sparse Mode Dense Mode} |
| PLMR | Previous Local Multicast Router |
| PPP | Point-to-Point Protocol |
| PSBU | Prefix Scoped Binding Updates |
| RA | Router Advertisement |
| RAME | Resource Access Mediation Entity |
| RCoA | Regional Care-of-Address |
| R.F. | Radio Frequency |
| RFC | Request For Comments |
| RP | Rendezvous Point |
| RPF | Reverse Path Forwarding |
| SAP | Session Announcement Protocol |
| SAS | Service Access Server |
| SDP | Session Description Protocol |
| SI | System Information |
| SIP | Session Initiation Protocol |
| SLA | Service Level Agreement |
| SSL | Secure Sockets Layer |
| STB | Set-Top-Box |
| TCP | Transmission Control Protocol |
| TLS | Transport Layer Security |
| TS | Token Server |
| UDP | User Datagram Protocol |
| UMTS | Universal Mobile Telecommunication System |
| UR | User Registry |
| USB | Universal Serial Bus |
| USR | User and Services Register |
| UTRAN | UMTS Terrestrial Radio Access Network |
| WEP | Wired Equivalent Privacy |
| WLAN | Wireless Local Area Network |
| Xcast | Explicit Multicast |
| Xcast+ | Explicit Multicast Extension |

Executive Summary

The European research project OverDRiVE (Spectrum Efficient Uni- and Multicast Services Over Dynamic Radio Networks in Vehicular Environments) aims at UMTS enhancements and coordination of existing radio networks into a hybrid network to ensure spectrum efficient provision of mobile multimedia services. An IPv6 based architecture enables interworking of cellular and broadcast networks in a common frequency range with dynamic spectrum allocation (DSA). The project objective is to enable and demonstrate the delivery of spectrum efficient multi- and unicast services to vehicles. OverDRiVE issues are: (i) improve spectrum efficiency by system coexistence in one frequency band and DSA, (ii) enable mobile multicast by UMTS enhancements and multi-radio multicast group management, and (iii) develop a vehicular router, that supports roaming into the intra-vehicular area network (IVAN).

This deliverable gives a description of all the demonstration activities in the project. Demonstration activities were carried out by workpackage 2 (Mobile Multicast) and workpackage 3 (Mobile router and IVAN management). The goal of the demonstration is to validate key concepts developed within the project and to present the results to a bigger audience, e.g. conferences and workshops. The demonstrator(s) were developed in evolutionary steps which is also reflected within this document, describing the certain stages and the setup at the events they were presented. At the end of the project work was devoted towards an overall demonstrator to show the combined concepts of workpackage 2 and workpackage 3. This demonstrator was successfully demonstrated at the HyWin 2003 workshop and the annual project review 2003 in Torino, Italy. The project demonstrated the combined usage of WLAN, UMTS, GPRS and DVB/T for IPv6 uni- and multicast traffic to moving networks including a mobile router. The concepts were visualized using dedicated applications like software download to cars, adaptive video streaming over heterogeneous networks and mobile multicast video streaming. Besides the “car” demonstration the concepts of optimized mobility management for large moving networks (combination of micro- and macro-mobility approaches) and group management for mobile multi-radio multicast to optimize network efficiency and user satisfaction were demonstrated.

The demonstrations provided valuable input to the project in order to validate the concepts and to identify further working areas. Showing the demonstrators at several events increased the overall awareness of the project in the scientific community.

1 Introduction

The European research project OverDRiVE (Spectrum Efficient Uni- and Multicast Services Over Dynamic Radio Networks in Vehicular Environments) aims at UMTS enhancements and coordination of existing radio networks into a hybrid network to ensure spectrum efficient provision of mobile multimedia services. An IPv6 based architecture enables interworking of cellular and broadcast networks in a common frequency range with dynamic spectrum allocation (DSA). The project objective is to enable and demonstrate the delivery of spectrum efficient multi- and unicast services to vehicles. OverDRiVE issues are: (i) improve spectrum efficiency by system coexistence in one frequency band and DSA, (ii) enable mobile multicast by UMTS enhancements and multi-radio multicast group management, and (iii) develop a vehicular router that supports roaming into the intra-vehicular area network (IVAN).

The work presented within this document was performed within the workpackages that deal with demonstration (WP2 and WP3). The objectives of the demonstration are to validate the main concepts and to show the major findings and concepts of WP2 and WP3 to a bigger audience (e.g. on workshops and conferences). The focus of the demonstration was on mobility management approaches suited for the OverDRiVE scenarios and requirements and on group management for mobile multicast.

The section following the introduction gives details on a demonstrator scenario and a non-technical story to raise the interest and understanding of the reader for the OverDRiVE vision. The demonstrator activities evolved over time during the project lifespan, and this is reflected in section 3, which gives a description of the several stages of the demonstrator and describes its evolution. According to the project plan and the project partners planning the demonstrator activities were focused at the end of the project towards a common / overall demonstrator approach. This approach, the architecture, and other aspects are described in section 4. The results of that activity are the demonstration at the HyWin 2003 workshop and during the annual project review 2003 in Torino, Italy. A detailed description for the both focus areas mobility management and group management is given in the following sections. Details of the setup, the explanation of the functional behavior and references to the according technical deliverables of the project give the reader an understanding of the work done for setting the system up. The actual services which made use of the concepts and that were presented to the audience to visualize the OverDRiVE achievements are described in section 7. The services and applications range from web access, to remote software update of car internals and to adaptive video streaming over heterogeneous access networks. The last two sections describe the external and internal interfaces that were used to connect to certain networks and systems.

The annexes give a description of details concerning porting of application to iPaq handhelds and some installation instructions for the software developed within the project.

2 Mobile Router Scenario and Demo Story

“Seamless connectivity and moving networks”

When Alice leaves her apartment she usually has already started downloading some of her emails to her mobile device and reading them. This morning the weather was hotter than expected and her car was parked in the sun. Therefore, Sandra has already remotely accessed the car to open the windows. Her son likes to watch a streaming video on his own mobile device in a WLAN HotSpot, while walking to the car. When entering the car, both mobile devices use the IVAN of the car for continued Internet connections when leaving the WLAN hotspot. For that reason they switch from WLAN connection to a Bluetooth connection inside the IVAN. The car utilizes high speed GPRS/UMTS and DVB/T for Internet connectivity. While the son keeps watching to his video, Alice downloads the remaining emails and starts the voice output function. Some time later she gets interrupted by an car software upgrade announcement from her car dealer. The software update will improve the remote access functions. Alice acknowledges the inquiry and the software download is scheduled immediately.

“This part will not be demonstrated:

After she has dropped her son at school and parked the car, a business video phone call arrives. She accepted as voice only call while hurrying to catch her train. Inside the train, she makes use of the higher bandwidth and switches to video mode. After she has finished this call she starts to synchronize her laptop with her business documents. They have been modified during the night from her team colleagues living in other time zones. After synchronizing she starts reading the modifications. Before Alice arrives at the executive workshop, she has read all latest modifications and is now well prepared for her presentation.”

At the end of the busy day the newly downloaded car software allows Alice to check the status of the car. She realized that she forgot to dim the interior light. She dims the light remotely by using her mobile device.

2.1 Technical Description

In this section we intend to describe the behavior of the components and protocols during our demonstration scenario in more detail.

When Alice leaves her apartment, her mobile device is connected in a WLAN hotspot, which is her home network, i.e. her home agent is located inside this WLAN hotspot (see Figure 1). The home agent is not involved in data transmission duties, since the PDA is reachable with its home address. Alice son’s PDA has identical settings, i.e. it is connected to its home network. The video server is located in the same network.

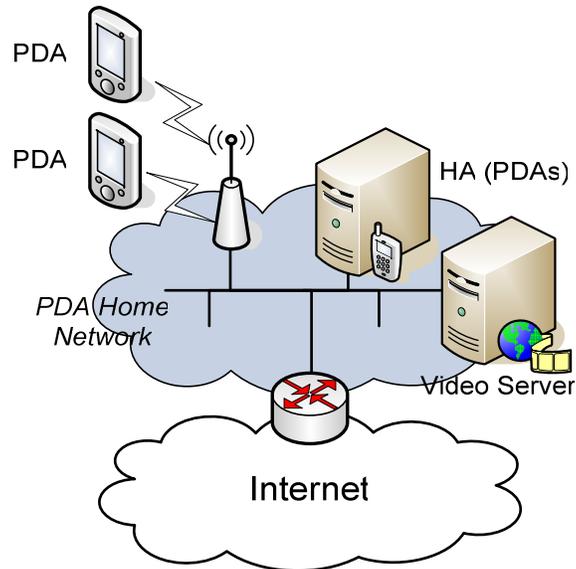


Figure 1: PDAs inside their Home Network

With her PDA, she queries the vehicle’s web server and opens the windows of her car.

When entering the car, both PDAs can switch from WLAN connection to either a Bluetooth connection or the IVAN WLAN connection. The PDAs detect their new network connection by sensing Router Advertisements (RAs) sent by the MR. They configure CoAs using the prefix information of the RAs. Both send Binding Updates to their home agent (i.e. the HA of the PDAs) to register their current COA and initiate message forwarding. The HAs accept the Binding Update and return Binding Acknowledgements.

Note that at this moment, the vehicle’s IVAN is already outside their home network. This means, there is already a tunnel established between the MR and the HA of the MR. As a consequence, the PDAs traffic is tunnelled twice, as depicted in the following figure.

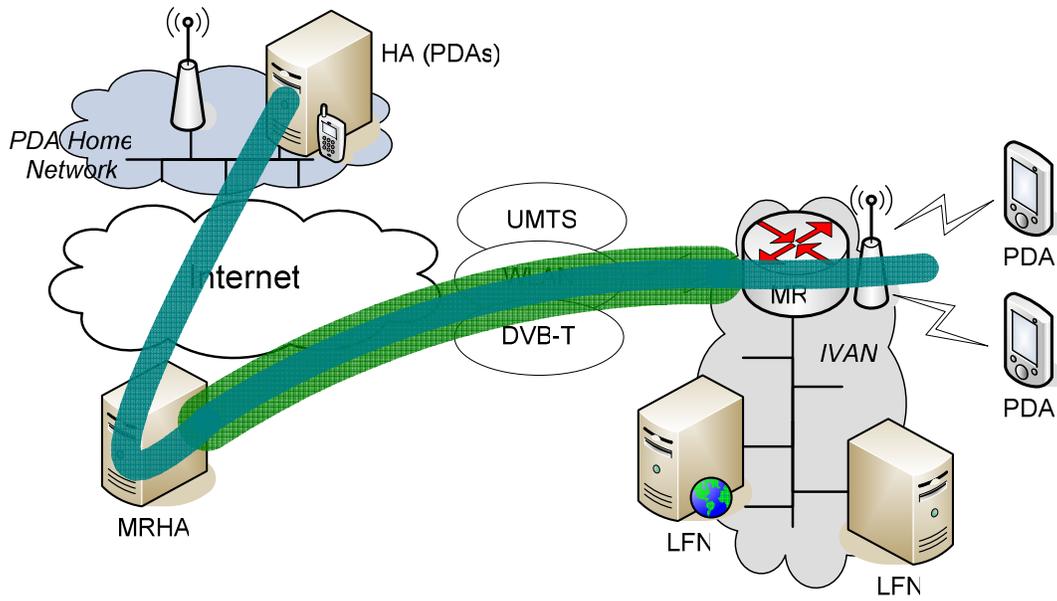


Figure 2: Nested Tunnels

The email download is now transmitted over GPRS/UMTS, while the son’s video is transmitted over DVB/T. To continue the session, the HA of Alice forwards all messages from the CN for Alice’s PDA to the COA of the PDA. In the MR home network, this COA is intercepted by the HA of the MR, since it is pretends to be the next-hop router for the respective subnet. The HA of the MR encapsulates it and sends it to the COA of the MR, which decapsulates it and forwards it to the PDA, which makes the final decapsulation.

The car software update is more simple than the PDA scenario, since it encompasses only a single tunnel, the tunnel from the HA of the MR to the MR. The MR decapsulates the software update and sends it to the LFN, in more detail the web server located on the LFN.

3 Demonstrator Development Description

3.1 Introduction

During the project duration all partners in workpackage 2 and workpackage 3 were continuously working on the development of the demonstrator. Following a discussion of a demonstration story and demonstration complexity the project developed the demonstrator(s) in evolutionary steps. The following section gives a description of the different stages of development and describes the evolution of the demonstrator.

3.2 Theoretical Overview of Ericsson Traffic Lab’s OverDRiVE Moving Network Testbed

Figure 3 shows the theoretical overview of the Mobile Router – Home Agent bidirectional tunnelling mechanism and the BRAIN Candidate Mobility Management Protocol combined solution. The Mobile Router – Home Agent tunnelling proposal is based on a bidirectional IPv6 tunnel between the mobile router (MR) and its home agent (HA). This tunnel connects the MR and its HA through the Internet and through different access systems. If the MR changes access system (ACS), the tunnel is torn down at the old ACS and is built up through the new one. The concept of this solution is described in [3].

In Traffic Lab’s solution either Visiting Mobile Nodes (VMN) or Local Fixed Nodes (LFN) can be attached to the moving network. The visiting mobile nodes can handle MIPv6 and BCMP signalling, but local fixed nodes are unaware of any kind of mobility (neither MIPv6 nor BCMP).

In the case of VMNs the mobile nodes get their IP address from the BCMP user registry, when they connect to the infrastructure inside the moving network, which is called Intra-Vehicular Area Network (IVAN). All IP addresses assigned to the visiting mobile nodes inside the moving network point to the BCMP ANchor Point (ANP), and are from the address space of the mobile router’s home agent. This way if a correspondent node from somewhere on the Internet sends packets to the mobile node that resides in the moving network, then according to Mobile IPv6, the home agent of the mobile node catches the packets. The home agent sends the packets to the mobile node’s care-of-address, which in this case points to the mobile router’s home agent. The mobile router’s home agent injects the packets into the MRHA tunnel and at the other end of the tunnel the mobile router forwards the packets to the BCMP anchor point, which in our case is co-located with the mobile router. From this point on, BCMP handles the delivery of the packets to the mobile nodes inside the moving network. Thus the anchor point tunnels the packets to that BRAIN Access Router (BAR), which the visiting mobile node is located at. The BAR sends the packets, this time without tunnelling through its radio interface to the destination mobile node.

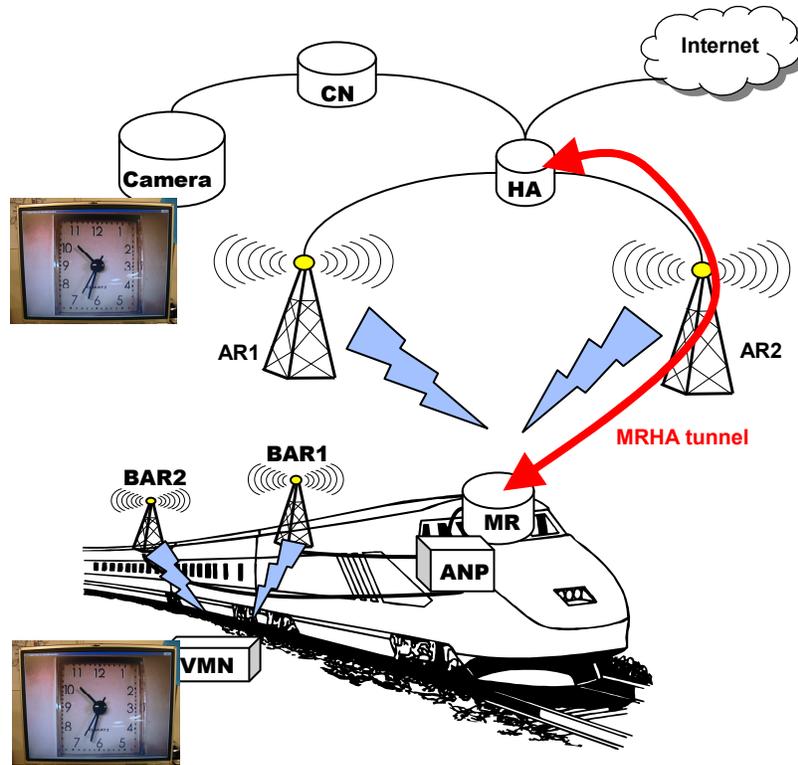


Figure 4: Ericsson’s OverDRiVE testbed overview (IST Mobile Summit 2003)

The overview of the Mobile Summit 2003 Ericsson testbed setup can be seen in Figure 4. A camera is connected to the Correspondent Node (CN), which streams a video of a clock in UDP packets to the home agent of the mobile router. The home agent injects the streaming content into the MRHA tunnel. The mobile router receives the UDP packets and passes them to the anchor point. The anchor checks, which BAR is the destination mobile node currently located at and tunnels the packets to this BAR. Finally the video of the clock can be seen on the mobile node’s display.

During the demo two different handovers (HO) were performed. The mobile node performed a BCMP handover between BAR1 and BAR2 in every second. These handovers were initiated by the mobile node and were performed with BCMP signalling. BCMP handovers did not influence the picture of the clock, which means that these handovers were performed very fast.

The second type of handover was performed by the mobile router between AR1 and AR2 once in every 6 seconds. This handover was initiated by a script that tore down the radio interface of the AR (either the radio interface of AR1 or AR2) which the mobile router was currently connected to. After the mobile router lost its connection it performed movement detection: first it sent a neighbour solicitation IP message (NS) to the old AR, but it got no answer since the old AR was already down; after a while it sent a router solicitation message (RS) to find neighbouring routers; after a successful reception of a router advertisement (RA) from the new AR it switched to the new access router and sent a binding update (BU) through the new link to its home agent. Thus, during this handover the MRHA tunnel is torn down at the old AR and built up through the new one. The radio interfaces of the access routers (AR1 and AR2) were configured to the same 802.11 channel. This type of handover takes quite a long time (1-6 seconds).

In Figure 5 the network topology can be seen. In the demo Ericsson used 8 network entities, a USB camera and a clock. The correspondent node was connected to the home agent directly with cable as well as the two access routers of the fixed network infrastructure. The home agent had 4

interfaces and on the 4th interface it was connected with cable to the IPv4 Internet. An IPv4/IPv6 web proxy, which was configured by University of Bonn was running at the home agent, thus IPv4 web connection was provided to the network entities.

The mobile router was connected through its 802.11 radio interface to the access routers, either to AR1 or AR2. Two access routers (AR3, AR4) were connected with cables to the mobile router. On these ARs BCMP was running, thus this access routers played the role of BARs. The mobile node was connected with its 802.11 radio interface to the BCMP access routers. A BCMP anchor point and a user registry were co-located with the mobile router. These entities handled the local mobility management inside the moving network.

As it can be seen in the figure, the mobility management of the entire moving network is solved with the MRHA bidirectional tunnelling mechanism based on Mobile IPv6 and the local mobility management inside the moving network is solved with BCMP. The scenario shows how a macro mobility (Mobile IPv6) protocol based network mobility approach can interwork with a micro mobility solution (BCMP).

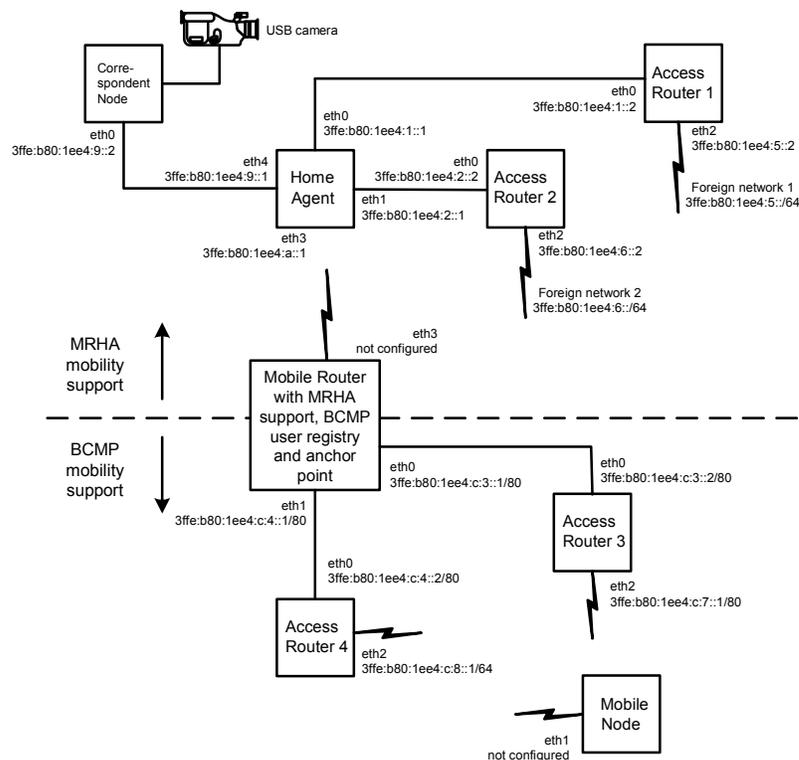


Figure 5: Network topology of the Ericsson testbed (IST Mobile Summit 2003)

3.3.2 CRM Demo

At the IST Mobile Summit in Aveiro, Portugal, June 2003, a simple moving network was demonstrated; it used two Wireless LAN Access Systems deployed at the conference center as well as the GPRS Access Systems publicly deployed by the TMN and Orange mobile communications operators. The demonstration featured a live webcam continuously sending images from the Motorola laboratory in Paris, France, while the Mobile Router was switching between WLAN and GPRS. The entire demonstration was based on IPv6, even if all access systems provided only IPv4 type of connectivity; issues raised by this exclusively IPv4 access

were solved by employing appropriate tunneling technologies. Also, the exclusively IPv6-enabled nodes inside the moving network could access the IPv4 World-Wide Web via a v6-to-v4 proxy server deployed at the University of Bonn.

The OverDRiVE demonstration was reported in the IST Results Bulletin, <http://www.istresults.info/index.cfm?section=news&tpl=news&ID=1403>

At the IST Summit conference in Milano, Italy, October 2003, Motorola performed a similar IPv6 moving networks demonstration jointly with CEFRIEL, Italy. At this demonstration, the video streaming source was managed by CEFRIEL and placed in the CEFRIEL laboratory (instead of using the source in the Paris lab). The Milano demonstrator featured three locations: (1) the access systems at the conference booth (WLAN hotspot and GPRS Telecom Italia), (2) the home agent in Paris and (3) video streamer was placed at the CEFRIEL labs (also in Milano but remote with respect to the conference booth).

In addition to the two IST demonstrations, CRM performed several internal demonstrations with the moving network testbed during the following events:

- Motorola CTO visit August 2003;
- Motorola Saclay Grand Opening / French Research Day, October 2003, Saclay, France;
- European Press visit, December 2003.

3.4 CRM Field Trials in Paris

These experiments expose different degrees of various problems, some not being directly related to the network mobility support itself. For example, the fact that deployed access networks only offer IPv4 access pose a significant problem in using Mobile IPv6; at the same time only a large-scale testing helps surfacing the significant gains that Route Optimization techniques can bring to basic network mobility support. None of these aspects can be studied in a laboratory configuration, field trials are needed.

In the following, we give an overview of the different logical components of the system, described in terms of IP networking:

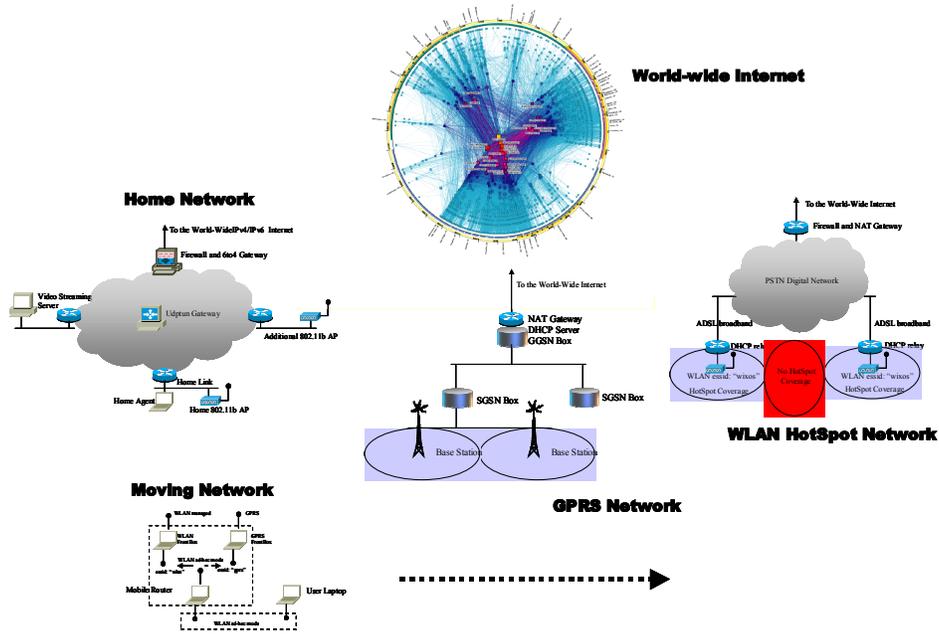


Figure 6: Overview of Field Experiments with a Moving Network

The world-wide Internet, pictured in the top diagram, is represented by a schematic set of inter-domain IP routes that connect various Autonomous Systems⁵. The important aspect to notice is that this mesh of routes must not be influenced by the moving network changing its attachment point. If the mobility protocol were to involve dynamic route updates at each movement, the effect on the mesh would be by and large catastrophic in terms of convergence times, not to mention the impossible-to-satisfy requirements on the size of routing tables. In order to avoid rendering the mesh routing unstable, the main Mobile IP requirement was to use a bi-directional tunnel between the mobile entity (MH and/or MR) and its Home Agent, thus relieving the need to propagate route updates necessary to maintain topologically correct addresses. Moreover, preserving the stability of the core Internet routing mesh and the implicit avoidance of routing interactions with the core network is a strong requirement when considering Route Optimization enhancements as well.

3.4.1 Moving Network

The structure of the moving network used in the field experiments differs from the moving network used in the laboratory experiments in a couple of ways:

⁵ Courtesy UC Regents, see Cooperative Association for Internet Data Analysis, CAIDA.org.

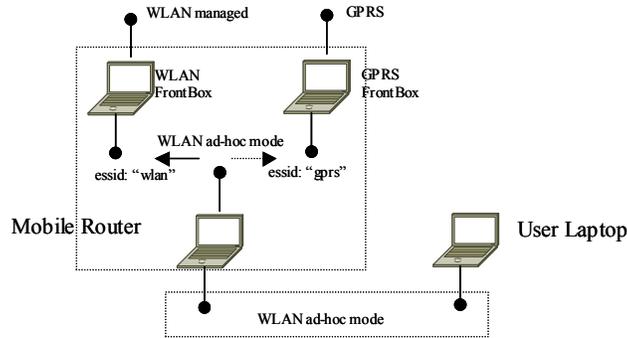


Figure 7: Moving Network

First, due to the space constraints (e.g. deployment in the trunk of a car) a convenience requirement indicated that no cable should be used, Ethernet or otherwise. Thus, whereas the moving network of the lab experiments was using a couple of Ethernet cables and a hub to connect together the Mobile Router and the Local Fixed Node, for the field experiments we used WLAN cards in ad-hoc mode for the fixed segment of the moving network. The entire moving network was not using any cable, only WLAN cards. While travelling, all laptops were working on battery power. Second, due to the impossibility to obtain native IPv6 addresses for the egress interface on the Mobile Router when attaching to access networks, new entities were designed to offer native IPv6 access to the MR: the FrontBoxes.

The Mobile Router depicted in the figure above has two WLAN cards. The first switches attachment between the two WLAN accesses offered by the WLAN FrontBox and the GPRS FrontBox. A Graphical User Interface on the MR offers the possibility to perform this switch by configuring different ESSID's and WiFi encryption keys, each corresponding to the one offered by the corresponding FrontBox. The second WLAN card is connected in ad-hoc mode to the peer WLAN card that is attached to the LFN laptop. The Mobile Router runs the LIVSIX IPv6/Mobile IPv6 stack.

The GPRS FrontBox is a laptop with one WLAN and one GPRS PCMCIA card. It uses the GPRS card to connect to the GPRS network and the WLAN card to offer native IPv6 connectivity to the Mobile Router. Similarly, the WLAN FrontBox uses one WLAN card to connect to the WLAN HotSpot Network and another WLAN card to offer native IPv6 connectivity to the Mobile Router. The two IPv6 prefixes advertised by the FrontBoxes to the Mobile Router allow it to form a different Care-of Address per access system, thus triggering native Mobile IPv6 management.

A FrontBox obtains, when in covered areas, an IPv4 address of the form 10.x.y.z by using the DHCP protocol. It also uses the IPv6 routing advertisement daemon radvd to advertise an IPv6 prefix to the Mobile Router. The ESSID identifier used by its WLAN card is different for each FrontBox, and with a different encryption key, allowing the Mobile Router to switch seamlessly (independent of to the IP layer) between each FrontBox.

The concept of a Frontbox not only helps with performing IPv6-in-IPv4 tunnelling but offers several additional benefits: (1) management of GPRS connections independently of the Mobile IPv6 exchanges performed by the Mobile Router, (2) isolation of mobility protocols from reliability-enhancement and seamlessness mechanisms (3) clear and simple event observability giving tight control opportunities on various mobility events such as ppp connexion re-launching, or manually switching between access systems and, finally, (4) opportunity to design and develop an independent DVB-T frontbox dealing separately with the complex behaviour of uni-directional (terrestrial or satellite).

However, the FrontBox concept is intended to be ephemeral, for testing purposes only. Once enough experience is gained, protocol behaviour is understood and corresponding software is reliably offering IPv6 access, all the FrontBoxes can be combined with the Mobile Router in a single box. Ideally, the Mobile Router box should have a number of wireless cards and antennas each connecting to various access systems (GPRS, WLAN hotspot, DVB, satellite) and an additional set of cable interfaces (Ethernet, Car Area Network CAN) and wireless cards and antennas (802.11b, Bluetooth) connecting to the inner subnets of the moving network:

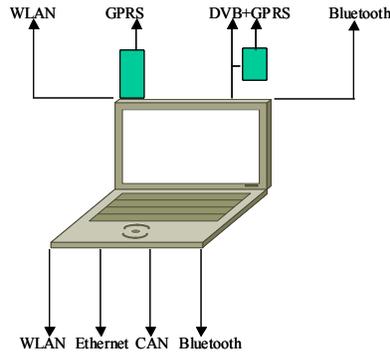


Figure 8: Envisioned Mobile Router

3.4.2 Home Network

The home of the moving network is described below. We used the same testbed network deployed in the local Motorola laboratory that was used for the laboratory experiments (as described previously in section 3.3.2).

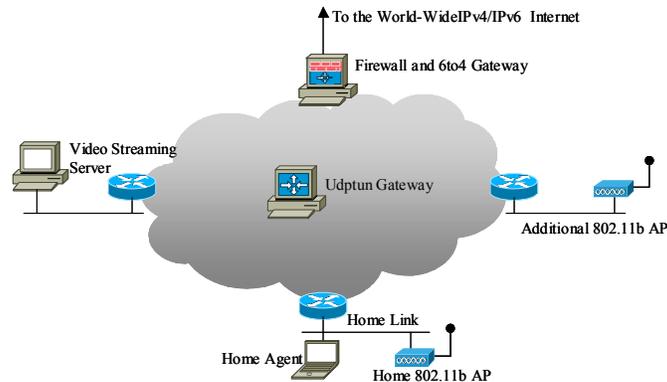


Figure 9: Home Network

The home link, pictured at the bottom, is served by a Cisco router and is an Ethernet segment, at 100Mbps speed. The Home Agent HA is an IBM laptop clocked 1.2Ghz and with one Ethernet 100Mbps interface. HA runs the LIVSIX IPv6 stack. Connected to the same home link is an 802.11b Access Point. The moving network is initially attached to this AP. Another similar IPv6 edge network is pictured at the right of the cloud, where an additional Access Point is connected. The third edge network, pictured at the left of the cloud, hosts a PC video streaming server, playing the role of a Mobile IPv6 Correspondent Node (CN). All three edge networks are assigned different IPv6 prefixes with length 64; all IPv6 prefixes in the testbed are aggregated under a unique 48-length 6to4 prefix. Note that a mobile entity moving between the first two subnets obtains different Care-of Addresses.

The testbed network is connected to the world-wide Internet with a Gateway that has firewall functionalities as well as IPv6 connectivity (Firewall and 6to4 Gateway). The 6to4 technique is used to derive the 48-length prefix from one public IPv4 address. Thanks to the generosity of such a short prefix, the numerous entities in the testbed are each assigned an IPv6 addresses (many non-relevant entities are not pictured), even if the home network is only assigned an IPv4 class C address range (only 255 entities are addressable).

Finally, the Udptun Gateway is pictured in the middle of the cloud. The Udptun Gateway bears its name from an abbreviation of the UDP tunnelling software, used to maintain a UDP tunnel to the MR, through various NAT gateways. This is used to offer IPv6 connectivity to the mobile router when the latter can only obtain a non-publicly routable IPv4 address (instead of a native IPv6 Care-of Address).

3.4.3 GPRS Network

The GPRS Network is pictured below:

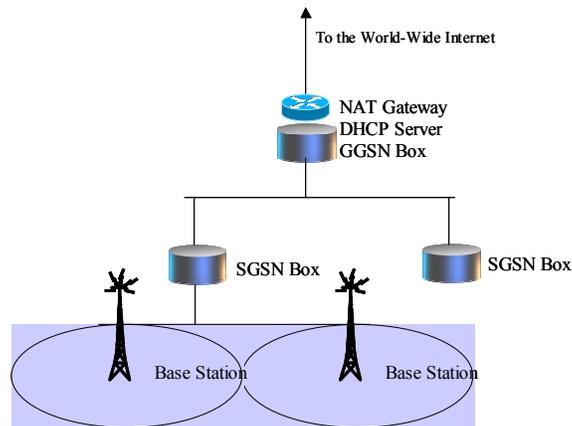


Figure 10: GPRS Network

The GPRS network used as an access system was Orange. Technical details of the specific topology of this network were not available, thus only a generic conceptual picture is given. It was observed that several Base Stations were deployed along a highway and that the corresponding cells were larger than cells used inside the metropolitan area (this was observed by visually supervising the power level sensed by a GPRS handset). It has also been observed that, when connecting to the GPRS network, a private non-routable address is obtained, of the form 10.1.12.x or 10.1.13.y. This implies that the GPRS network is connected to the world-wide Internet with one, or a set of, NAT Gateway(s). It should also be noted that the address is distributed with the DHCP protocol, thus a DHCP server is necessarily present in the GPRS network.

Establishing a tunnel between the GPRS FrontBox and the Udptun Gateway requires exact knowledge of the addresses and port numbers of each of the two endpoints. On the Udptun Gateway endpoint this is easily achievable since it is placed in the home network and under local control. However, the source address and port number on the GPRS FrontBox are not easily identifiable, even if it is under local control. The problem lies in the fact that even if the source address and port number can be known by consulting local variables, they are dynamically changed in each packet that traverses the NAT Gateway. Since the NAT Gateway is not under our control (part of the GPRS Network), properly identifying the source address and port number

corresponding to the GPRS FrontBox is a challenging task. See the section 3.4.5 on UDP Tunneller software about the way this is achieved.

3.4.4 WLAN HotSpot Network

The WLAN HotSpot network is pictured below:

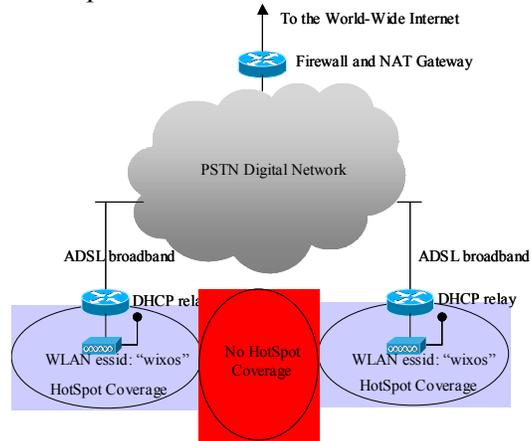


Figure 11: WLAN HotSpot Network

The WLAN HotSpot Network used was the experimental hotspot network Wixos⁶. A detailed description of this network was not available, so we only present a conceptual view of the network, basing our picture on some public descriptions offered by Wixos and some results of practical observation.

The behaviour of an IP host connected to a WLAN HotSpot Network is similar to that of a GPRS Network: an IPv4 non-publicly routable address 10.x.y.z is obtained and thus client-style⁷ access to the world-wide Internet is offered. A first important differentiator noticed is the high-bandwidth, in the range of between 5 to 7 Mbps (when compared to a 56Kbps max in the GPRS network). We assume that every hotspot area is featuring a Cisco Access Point with a router and a DHCPv4 server. We assumed that these two entities are connected with ADSL subscriber lines to the PSTN infrastructure.

Accessing the Internet when in a hotspot area usually gives satisfactory results, in terms of bandwidth, video streaming with popular tools such as RealPlayer is satisfactory. However, when trying to move from one hotspot area to the next, one faces difficult challenges like shadow zones and changes in the IP address.

Note that unlike the GPRS cell coverage, the hotspot areas are likely to be separated by large uncovered areas. GPRS cells are most of the time overlapping but hotspot areas are not. Hence the importance of exploiting the opportunity to change between access systems. When in a hotspot area high bandwidth is used, and when outside hotspot areas the GPRS coverage should be used even if with less bandwidth.

⁶ At the time of writing, the hotspot Wixos experimental network evolved into two separate commercial offerings, backed by two different companies: WIFISPOT and WIFIX .

⁷ The obtained private IPv4 address can be used to initiate sessions to other sites on the worldwide Internet, but other sites from the Internet can not initiate sessions to the holders of private addresses. This effectively limits the level at which a host connected to GPRS can participate in Internet sessions (e.g. VoIP, Instant Messaging and similar sessions are impossible).

In addition, an IPv4 address valid in one hotspot area is invalid in the nearby hotspot (topologically incorrect). The DHCP client software needs to be re-triggered when entering a new hotspot area.

3.4.5 UDP Tunnel Software⁸

Runs on both FrontBoxes and on the Udptun Gateway. Solves the problem of offering a native IPv6 Care-of Address to the MR, when the access system only offers a non-publicly routable address (DHCP NAT). Encapsulates IPv6 packets within UDP IPv4 packets. The IPv6 address offered to the Mobile Router allows bidirectional communication to the worldwide Internet and full reachability for session initiations: unlike the IPv4 address obtained on the Frontbox, both initiations from LFN to CN *and* from CN to LFN are allowed (such as VoIP and Instant Messaging is possible).

3.4.6 Mobility Scenario

In some of the high-level scenarios of the project Overdrive, a simple network mobility sequence was proposed as: first move the mobile network within the laboratory, then deploy it in a car, travel across a short 20km highway, join a hotspot area in the city and move to another hotspot area across a blind zone (in which GPRS would be used). From this high-level description, the following functional behaviour was deduced: (1) moving network first connected to the home link, (2) next connected to the additional AP in the home network, (3) then connected to the GPRS network, (4) then connected to the WLAN HotSpot, (5) connected again back to the GPRS Network and finally (6) connected to another hotspot WLAN. Having proposed this scenario, the effective testing was performed. Practical observations could be described with the help of the picture below:

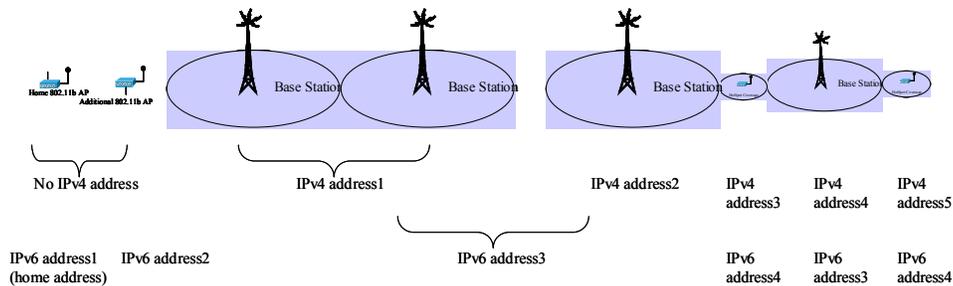


Figure 12: Mobility Scenario and Dynamics of IP Address Assignment

Scanning the above figure from left to right, one can initially notice the two Access Points in the home network, then several large GPRS cells along the highway are pictured. Some of the GPRS cells are separated by shadow uncovered areas such as a physical underground tunnel, or simply lost ppp connectivity due to interference or network overloading. Finally, within the city, two small hotspot areas are separated by a larger GPRS cell (not as large as the cells along the highway). The dynamics in the assignment of IPv4 and IPv6 addresses are shown below the corresponding cells/areas. The middle line pictures the assignment of the IPv4 address to each FrontBox respectively. The bottom line depicts successive changes of the IPv6 address assigned to the egress interface of the Mobile Router (from Home Address to different CoA's) a.

⁸ A detailed description of UDP Tunnel software for NAT traversal and in combination with the Mobile IPv6 protocol is not given here to the proprietary aspects of this information (MCP).

The IPv4 address is relatively stable when within the GPRS network and changing when in the WLAN hotspot areas. The GPRS FrontBox changed its IPv4 address while connected to the GPRS Network (IPv4 addresses 1, 2 and 4) while the WLAN FrontBox changed its IPv4 address too (IPv4 address 3 and 5). A different IPv6 prefix is advertised by each of the access points in the home network and by the FrontBoxes. The Mobile Router changed its IPv6 address from the home link (IPv6 address 1) to the additional access point (address 2), to the GPRS network (address 3) and to the WLAN network (address 4).

From an encapsulation standpoint, the end-to-end communication between LFN and CN dynamically evolved between three different typical phases. Each of these phases is pictured in the diagrams of the figure below:

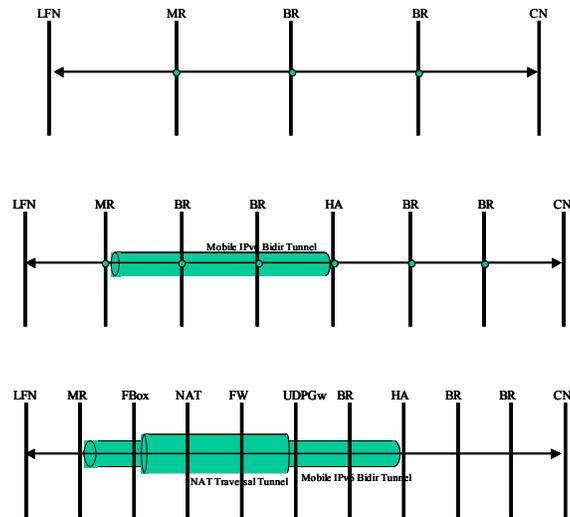


Figure 13: Tunnelling Dynamics during IP Mobility Scenario

The top diagram shows direct path communication between CN and LFN while the moving network is at home. Remark in this case that no tunnelling is used.

The second diagram pictures the case when the MR attaches to the additional home link (still in the home network). Remark in this case that the packets are encapsulated when travelling between MR and HA.

The third diagram represents the CN-LFN communication when the moving network is attached to the GPRS access system or the WLAN access system. Remark in this case a second level of encapsulation occurring between the respective FrontBox and the Udp Tun Gateway (encapsulation in a IPv4 UDP stream).

From a Route Optimization standpoint, the field experiments allowed identification of a very strong need of using optimal paths (bypassing the Home Agent). If one considers that the GPRS network hosts a CN, like a telecom provider website, then the path between LFN and that CN (when MR attached to the GPRS network) is very short, in the order of 3-7 IP hops. However, the HA is positioned very far in the home network, at more than 30 hops away. In this case, the difference in path lengths is intuitively an order of magnitude large. It is expected that optimal paths used in this case would bring in important benefits to application behaviour. Moreover, this scenario being the simplest possible, it might invite to think that route optimization enhancements to the protocol are relatively simple. However, richer configurations render such reasoning difficult to follow, since two MR's attached to WLAN and GPRS respectively can be literally

very close to each other⁹, thus RO gains can be thought of as potentially important (the number of IP hops between the two MR's may be in reality very large). The situation can be even further complicated if considering that certain future 3G networks consider deploying HA within the GPRS network, thus the HA and CN will be relatively close to the MR (and not far away in the home network, as in our experiments); in that case, again, benefits of using shortest paths might not be very important.

3.5 OverDRiVE Project Meeting Budapest

At the Budapest meeting ETH and EAB-BUTE partners presented two demos with their integrated testbeds. The integrated testbed consisted of EAB-BUTE mobile multicast testbed and ETH's moving network testbed, which got some extensions since the Mobile Summit demo. With the integrated testbed both multicast and unicast packet delivery and multicast and unicast seamless handovers were demonstrated.

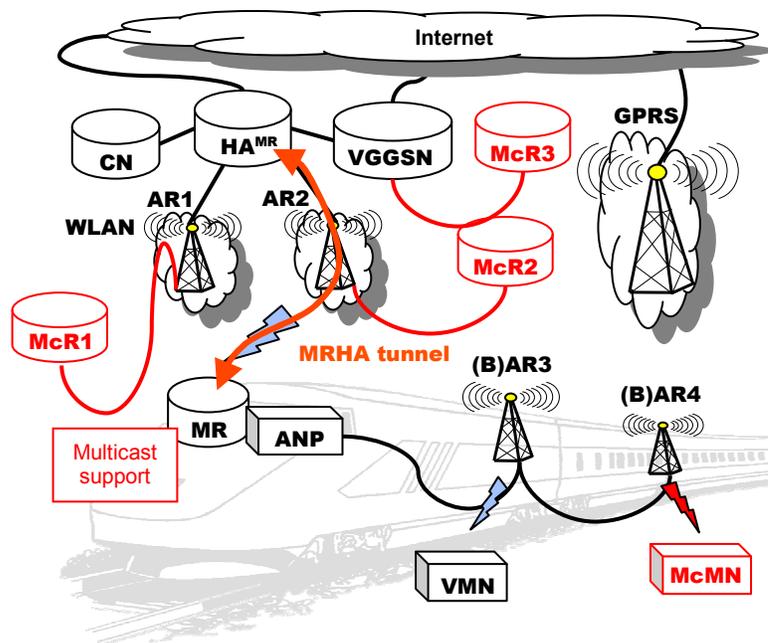


Figure 14: Ericsson OverDRiVE testbed overview (Budapest meeting)

The integrated testbed can be seen in Figure 14. This testbed setup differs from the Mobile Summit testbed setup (Figure 4) in the followings:

⁹ The GPRS and WLAN HotSpot networks are relatively close in terms of physical distance: in a city, one often has visibility to both a GPRS Base Station and a WLAN Access Point antenna. However, in IP terms, this physical closeness is irrelevant. On one hand, even if the obtained IPv4 address from each system has the same form 10.x.y.z (and sometimes can even be the same e.g. 10.1.2.3), these addresses are most certainly not connected by a short IP path. The two networks are entirely different and independent, their first meeting point being somewhere on the worldwide Internet. Bottom line is that the length of the direct IP path between the two addresses obtained at the same physical place (the number of IP-addressable hops) is quite large. This paradox of physical closeness but large distance in IP terms gives an intuitive example of the complexity of the aspects that should be paid attention to when considering Route Optimization enhancements.

- GPRS support is added to the testbed and a VGGSN is connected to the home agent of the mobile router. To understand the necessity of VGGSN we shortly explain how the GPRS connection of the mobile router is configured. To reach the mobile router from the Internet it has to have a publicly available and routable IP address. When the mobile router is connected through GPRS it gets the IP address from the service provider of the access system. In most cases the operators only give addresses of their private domains (e.g. 10.x.y.z), which are not available from outside. To allow users to reach the Internet these providers use native address translator (NAT), but NAT provides only one-way reachability, which means that it does not allow reaching the node from the Internet by anyone. To make a node inside a private domain available from outside the domain we have to build up a tunnel through the NAT between the node inside the private domain and another node somewhere in the Internet, which has a publicly available, routable IP address. Because of this NAT issue we had to employ an entity, which we called Virtual GGSN, since it could be regarded as an access point of the GPRS network. Although the functionality of this entity could be integrated in the mobile router's home agent as well, we decided to keep it separately, so we can point out that this functionality is not necessary to be run in the home agent of the mobile router. To solve the tunnelling through the GPRS access system we used VTun [17], but IPsec and FreeSWAN could be used as well. In this setup we also used a GPRS FrontBox, which was connected to the MR and was the endpoint of the tunnel through the GPRS, but later we integrated the functionality of this entity into the mobile router.
- The testbed is also extended with a fast handover support. To perform fast IP handovers we can tell the MIPL stack at the MR that the old AR is lost. After we tell this to the stack (we set one bit to 0), the stack will immediately switch to another available AR. This fast handover is absolutely seamless; it does not disturb the picture of the clock which is shown during the demonstration.
- The testbed has multicast extensions because it is integrated with the EAB-BUTE testbed. Multicast routers are connected to the VGGSN and to the WLAN access routers and a multicast mobile node is placed in the IVAN. The multicast mobile node could perform handovers between AR1 and AR2 and it could roam into the IVAN and connect to one of the access routers inside the moving network.

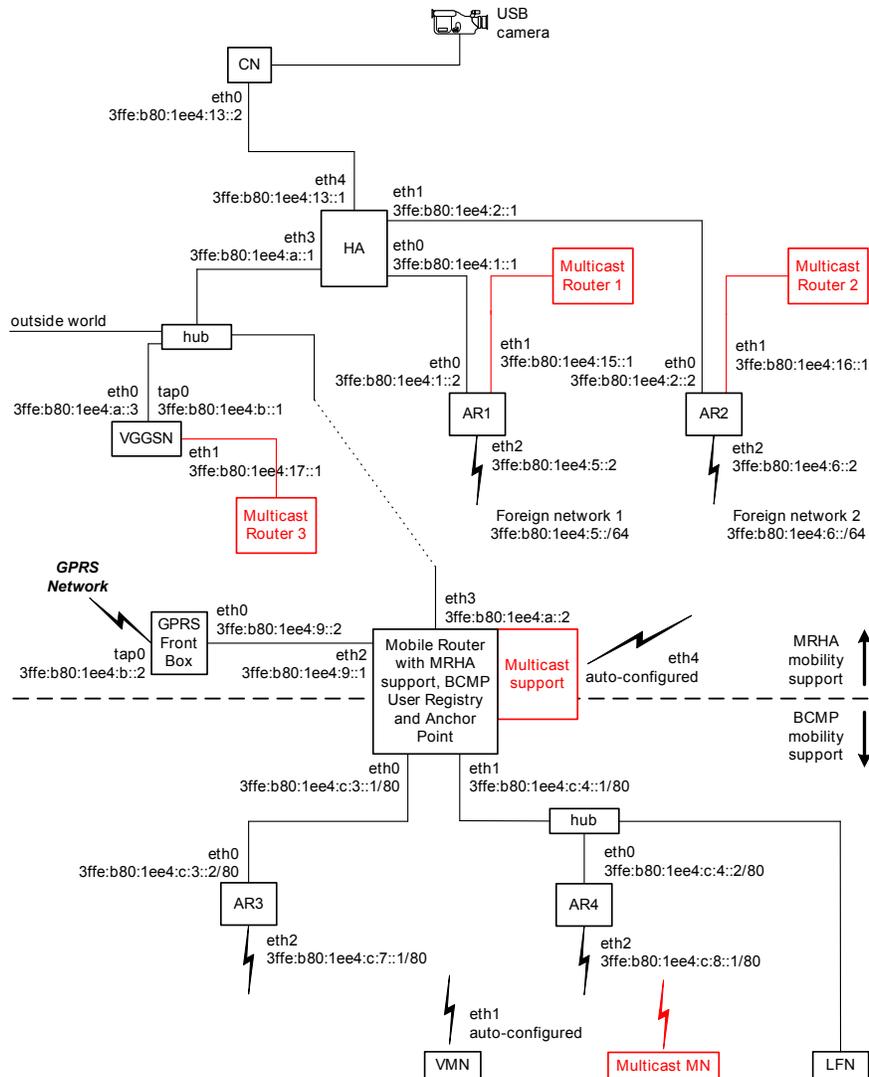


Figure 15: Network topology of the Ericsson testbed (Budapest meeting)

The network topology of this setup can be seen in Figure 15. We can see that the testbed setup is a bit more complex than the setup at the Mobile Summit (Figure 5).

3.6 PCC Wireless Communications Research Days

ETH presented its OverDRiVE moving network multi-access testbed extended with 3G mobile technology (UMTS/WCDMA) vertical handover and local fixed node (LFN) support at the PCC Wireless Communications Research Days (<http://www.pcc.lth.se>) in November 2003 in Stockholm (Kista Electrumssalen), Sweden. The visitors were mainly from Swedish universities ([Chalmers University of Technology \(CTH\)](#), [Royal Institute of Technology \(KTH\)](#), [Lund Institute of Technology \(LTH\)](#), [Uppsala University](#), and Luleå University of Technology) and also from Telia-Sonera and Ericsson. The demo has shown that UMTS/WCDMA provides the necessary bandwidth for real time video applications. In Stockholm, ETH used the publicly available UMTS/WCDMA connection of the 3G mobile phone service provider called “3”.

At this time the testbed had the following features implemented (the list below shows the evolution of ETH’s OverDRiVE testbed):

- BCMP handover of mobile node inside the moving network.
- MIPv6 handover between WLAN access routers, when the current AR disappears, then after MIPv6 movement detection the mobile router connects to another AR (first demonstrated at the Mobile Summit).
- MIPv6 handover between WLAN access routers, when the mobile router is told to move, it moves without movement detection. This is a very fast, seamless handover (first demonstrated at the Budapest meeting).
- GPRS connection, vertical handover between GPRS and WLAN (first demonstrated at the Budapest meeting).
- Mobile multicast support with the integrated EAB-BUTE testbed (first demonstrated at the Budapest meeting). This feature was not shown explicitly at the PCC Workshop.
- Local fixed node (LFN) support. Windows XP with IPv6 mobility support was installed on a laptop, which was unaware of MIPv6 and BCMP and this laptop was connected to the mobile router. Users using the laptop were able to browse the web using the IPv4/IPv6 web proxy configured by OverDRiVE partner UBN (the LFN support was first demonstrated at the PCC Workshop; web browsing was first demonstrated at the Mobile Summit).
- UMTS/WCDMA connection, vertical handover between GPRS, UMTS and WLAN (first demonstrated at the PCC Workshop).

The overview of the testbed can be seen in Figure 16. We can see that the testbed is extended with UMTS/WCDMA and LFN support.

Figure 17 shows the network topology of the testbed. It can be seen that a UMTS frontbox (UMTS FB) connected to the mobile router is used to reach UMTS. This entity was needed because the UMTS phone had only Windows XP drivers, and we had to install Windows XP on a machine to communicate with the phone.

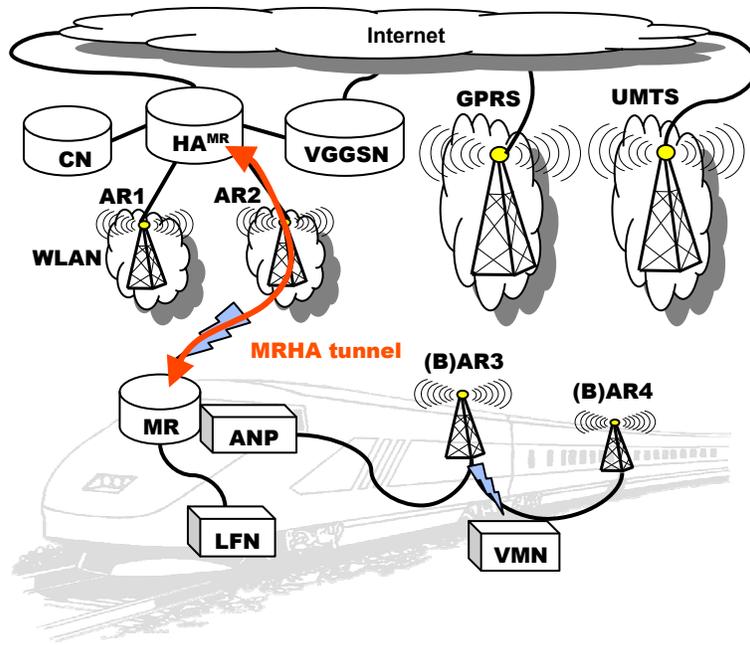


Figure 16: Ericsson's OverDRiVE testbed overview (PCC Workshop)

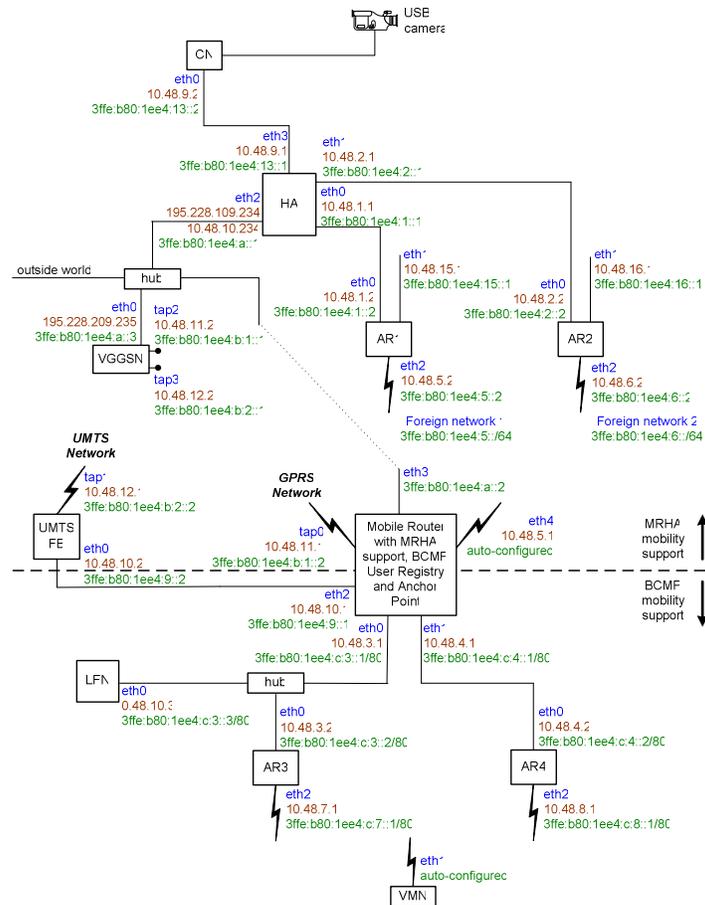


Figure 17: Network topology of the Ericsson testbed (PCC Workshop)

4 Common Demonstrator Architecture

The functional demonstration architecture is shown in Figure 18. Basically it can be divided into the core network which is native IPv6 and which in turn has also a connection to the world wide 6bone. The core network also provided connectivity for the content and streaming servers. Different access networks provide connectivity for mobile clients. In the demonstration the project utilizes DVB-T, GPRS/UMTS and WLAN. The moving network is connected via a mobile router with the access networks and provides inside the moving network further in-vehicular access networks like WLAN, Bluetooth and fixed Ethernet.

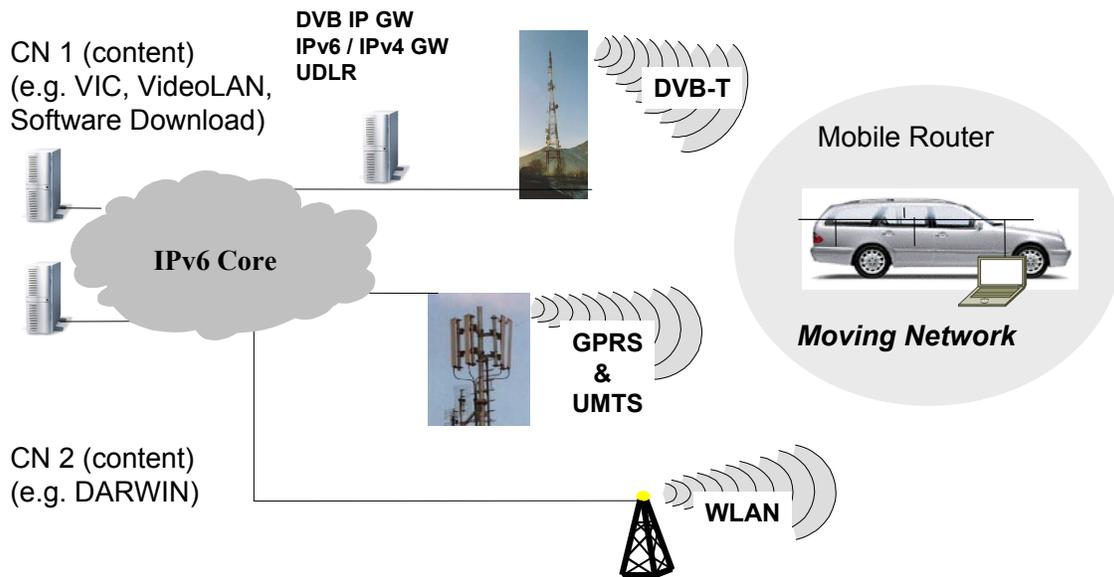


Figure 18: Common Demonstrator Architecture

4.1 Demonstrator Setup

A detailed overview of the demonstrator set up is shown in the following figure.

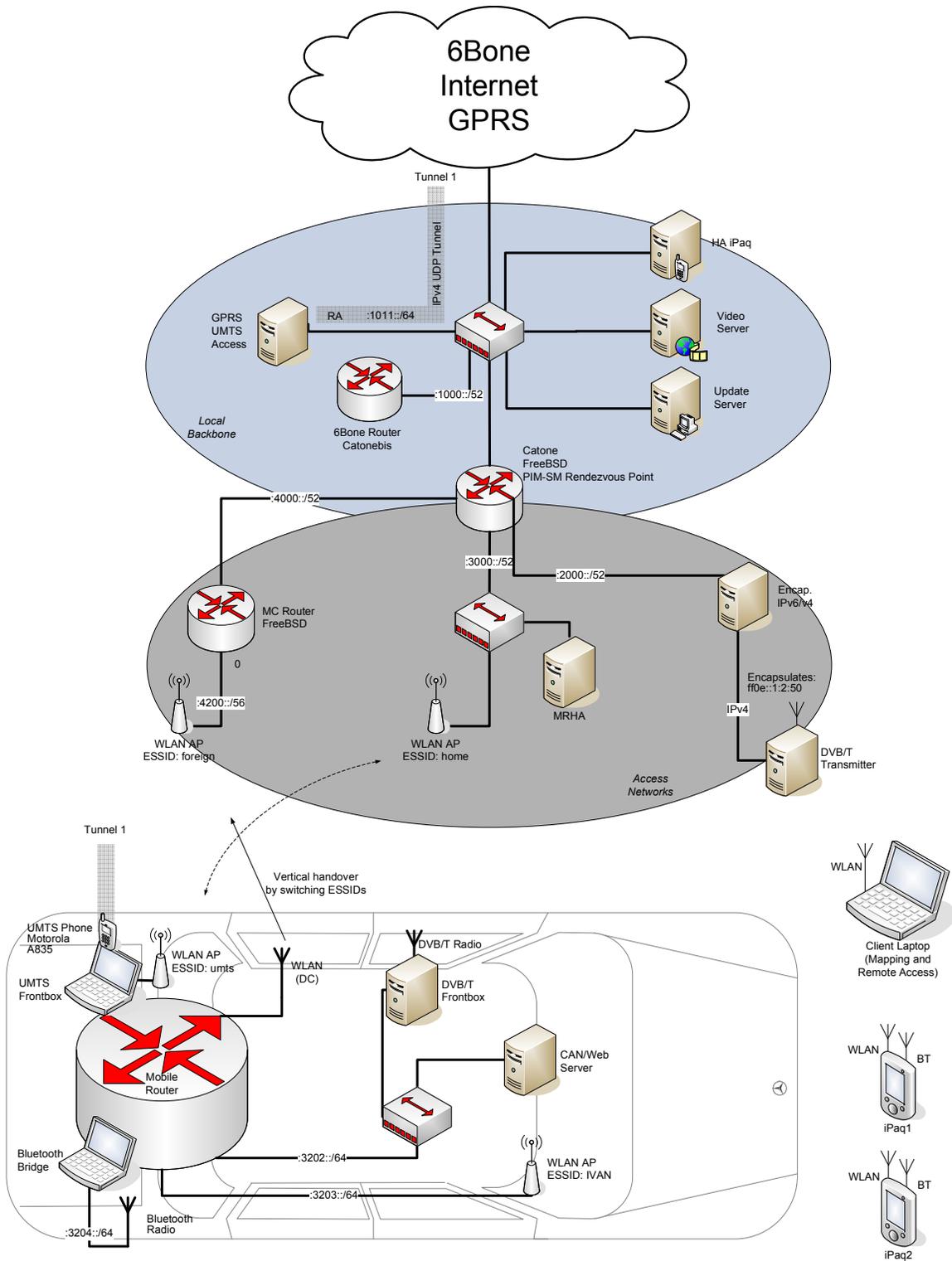


Figure 19: Demonstration Setup

We see that there are three external interfaces (WLAN, UMTS and DVB/T) and three IVAN internal interfaces (WLAN, Bluetooth and CAN). These interfaces will be described in Section 7 and Section 9 in more detail.

The basis of the demonstrator setup are presented in the following paragraphs. The MR handles application transparent IVAN mobility. The fixed hosts inside the IVAN are not affected by mobility. This means, mobility management is provided only by the MR instead of all devices within the IVAN. For our demonstrator set up, this includes the vehicle web server, the CAN server and possibly further fixed nodes which may be added to our demonstrator later.

The situation for mobile devices moving into the IVAN differs slightly, since they have to do an initial binding to the IVAN (i.e. register their IVAN COA at their HA once). Then further mobility management is done by the MR without affecting the mobile devices. In our demonstrator set up, a mobile device is given by the two PDAs, which moves into the IVAN.

The home agent of the MR is attached to the core IPv6 network and is stationary. This HA and the MR handle mobility on behalf of all LFN inside the IVAN. The HA of the mobile devices like PDAs may differ from the HA of the MR but of course is also connected to the IPv6 core. An overview of this functional components and their placement in the network is shown in the following figure.

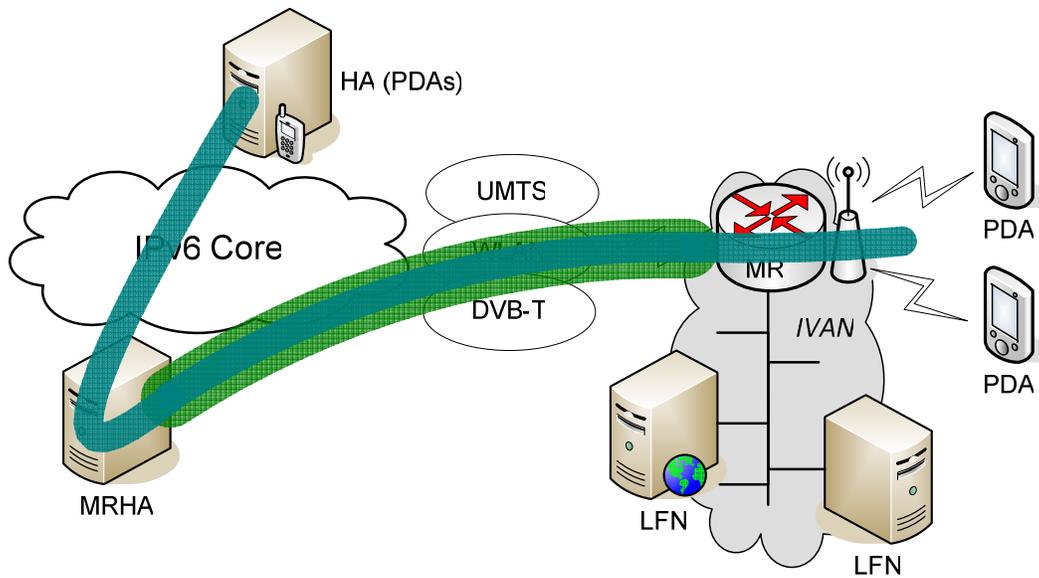


Figure 20: Functional Components

5 Mobility Management

Mobility management for the common OverDRiVE demonstrator includes mobility management for unicast sessions as well as for multicast sessions. Even though we offer a separate description for those two types of communications, the implementation basically uses the same IPv6 stack on several different entities: for unicast sessions it uses the Mobile IPv6 part of the LIVSIX stack (on MR and HA) while for multicast sessions it uses the MLDv2 part of the same stack (on the MR) accompanied by the MLD implementation of PIM-SM running on the Access Routers.

5.1 Mobility of IPv6 Unicast Communications

Mobility management for unicast communications is performed based on the Mobile IPv6 protocol for Mobile Hosts and Mobile Routers. The LIVSIX IPv6 stack is used on the Home Agent as well as on the Mobile Router. Neither the nodes within the moving network nor the Correspondent Nodes (video streaming servers, update servers and so on) are performing any form of mobility management.

5.1.1 LIVSIX IPv6 Stack

LIVSIX is an IPv6 stack designed entirely from scratch, with special regard to high portability and support for mobility environments. Portability means portability across operating systems and C compilers, portability in terms of code size, and portability in the sense of minimal modifications to existing IPv6 applications and to TCP. LIVSIX sources are available under the Motorola LIVSIX Public License and can be downloaded from the LIVSIX homepage, <http://www.enr.motlabs.com/livsix/>. Today, LIVSIX is developed as a Linux kernel module and meant to be used predominantly in mobility environments. A port to FreeBSD is planned. The LIVSIX architecture is separated into three abstraction layers:

- plt: platform dependent layer, kernel interface
- sfk: software framework and adaptation layer between plt and sfk
- csx: platform independent protocol implementations

Code modularity is enforced by considering the sfk part as a “thick interface” between csx and plt. All functions in csx or in plt can only call functions in sfk. The sfk is the only place where both functions from plt and csx, can be called. The csx layer holds the major part of the source code, e.g. the routing-related algorithms, the packet construction and the mobility management. The biggest part of the sfk-layer consists in wrapper functions between the other layers and the plt layer is responsible for the signal and buffer exchange between the LIVSIX module and the kernel. As a consequence, when porting to another operating system, the csx layer remains untouched, the sfk layer requires minor changes and the plt layer needs full adaptation. Due to the concentration on mobility environments, the first release of LIVSIX implemented only rudimentary host functionality, e.g. only partial support for UDP and no TCP. This enabled the usage of simple applications as ping6 or tftp. Subsequently, host mobility has been added as well as simple TCP support. Current development introduces router functionality, improved TCP and UDP support, and provides a mobile router implementation for mobile networks.

5.1.2 IPv6 Router

Router functionalities for the LIVSIX IPv6 stack are implemented in separate modules, to avoid too much modifications of the existing code. The router advertisement procedure is embedded in the existing code for neighbor discovery, the routing table, its management and the packet routing itself are located in a separate part. This separation guaranties independent development and test of the modules. According to the LIVSIX architecture, most of the code, that realizes the router implementation, is located in the platform independent layer. Only packet sending and receiving packets, as well as passing configuration options to the kernel module require platform dependent code and wrapper functions. The existing function that handles the packet reception is extended. Before dropping the packet, if routing is enabled, LIVSIX tries now to route the packet. This results in the following packet processing procedure:

1. packet reception as home agent (if enabled)
2. packet reception as host
3. packet routing (if enabled)
4. drop packet

As specified, router functionality is only enabled if the global `IsRouter` variable is set. Setting this variable causes LIVSIX to ignore router advertisements from other routers, as required for a router. The existing default router list is cleared and only routes that have been added administratively to the routing table remain valid. In other words, everything that has been learned through router advertisements from other routers is deleted. The router joins the all-routers IPv6 multicast address (`FF02::2`) and the all-routers Ethernet multicast address.

5.1.3 Routing Advertisement Module

Both the Linux kernel IPv6 stack and the KAME reference stack, do not implement sending router advertisements directly in the kernel. They make use of external router advertisement daemons (`rtadvd` or `radvd`). LIVSIX realizes an implementation in the stack itself. The decision not to use an external daemon was taken out of several reasons. First, since sending router advertisements is a part of neighbour discovery, and most functionality of neighbour discovery has to be implemented directly within the stack, it seems reasonable to implement the entire mechanism in the stack. Furthermore a stack implementation simplifies monitoring of router advertisements from other routers and the comparison to the own advertisements. And finally, the integration in the stack can be used to support special functionality for mobile routers (like blocking outgoing router advertisements on the mobile interface, while being connected to a foreign network).

The drawback of a kernel module implementation is the variety of possible configuration options that have to be passed to the module, to allow a RFC conform implementation, even though normally few of them are really used. However, having a kernel implementation does not prevent administrators to use external router advertisement daemons. To pass the configuration options to the kernel module, the Linux `sysctl` interface is used. Every option described previously is represented by an entry (this means, options that apply to an interface are represented once per interface, options for a prefix once per prefix). Since dynamic home agent discovery is not supported by LIVSIX, the related options are not included. Advertising a prefix is started as soon as the minimum set of options is configured. This set includes a prefix, the prefix length, the `AdvSendAdvertisements` variable on the interface and the `IsRouter` flag. All other values are initialized by the suggested defaults.

After setting the AdvSendAdvertisements variable to “true”, a timer is scheduled with a randomized value, as defined in the specification mentioned previously. Upon its expiration, an unsolicited router advertisement is sent and the next timer is initialized. Modified options are taken into account upon next timer expiration. Every interface is associated an individual timer, as soon as the AdvSendAdvertisements variable is set. Multiple prefixes on the same interface are aggregated to one advertisement. Currently, a maximum of ten prefixes per interface is defined and supported. The prefix advertisement data is stored in an internal data structure. As mentioned above, data that is not modified by the user is initialized with predefined default values. The data structure is organized as a list that contains one element for every interface. Within these elements, options for router advertisements are stored as well as a pointer to a list of prefixes and prefix related options that are advertised on the interface.

5.1.4 Routing Module

5.1.4.1 Routing Table Structure

LIVSIX is not meant to be used in a backbone network, but to be used in access networks. That means, future routing tables will hold approximately five entries, max. twenty entries. As a consequence, a simple table structure is sufficient. This leads towards the simplest implementation, a double chained unsorted list. Research in this table is performed by always examining the whole list to find the optimal route for a given destination. Every list element contains the fields specified in previously Nevertheless, to be able to switch to a different structure and more performant lookup algorithm, the table structure as well as the related functions are implemented well separated from other code, to be easily changeable.

5.1.4.2 Routing Table Management

The routing table is internally managed by the functions listed previously. Access from outside of the stack is provided by two interfaces, a sysctl interface and a clone of the Linux IPv6 stack ioctl interface. The ioctl interface was implemented to enable compatibility to existing applications, such as the “route” command or routing daemons, while the sysctl interface provides the advantage to be usable on every operating system, that supports a /proc filesystem. Both interfaces use the internal functions for modifications of the routing table. The current routing table is permanently accessible in “/proc/net/livsix_route” or “/proc/net/IPv6_route”, the latter is implemented for Linux IPv6 stack compatibility. Since the routing table fields “flags”, “metric” and “use” are not used or modified by the stack itself, the ioctl interface is the only way to modify them. The most important support function is the function to find a specific route in the routing table. Routes are identified by the destination prefix or address and its corresponding prefix length. Having two or more routing table entries with the same combination of both of these values is not allowed. The function browses the routing table for a given combination and returns a pointer to the entry, in case of it exists, or otherwise a zero pointer when reaching the end of the table. To add a new entry to the routing table, the responsible procedure requires at least the destination prefix, the prefix length, the gateway address and the device as parameter. Other fields of the table, that are omitted, are initialized by default values. The first task is to verify, that no other entry with the same prefix and prefix length exists, by using the above research function. If there is already such an entry, adding the new one is refused. If nothing is found, memory for a new table entry is reserved, filled with the data and linked to the existing routing table.

Deleting a route requires only the two parameters that are sufficient to identify and find the route, the destination prefix and the prefix length. The corresponding table entry is removed from the list and the associated memory is freed. The function that is used to create a new table initializes an empty table with a newly created entry.

Since all routing table modifications are done by using these four functions, switching to another table structure is possible by changing the contents of the functions (plus adapting the lookup function), while keeping the prototypes. No additional changes to the stack are required.

5.1.4.3 Packet Routing

Destination research in the routing table is done by a separate lookup function. The function requires a destination address as input parameter and returns the next hop address and the outgoing interface. The longest matching prefix is searched examining all routing table entries and comparing bitwise the destination address to the prefix. If the address matches the prefix, the table entry is kept as a possible destination, as long as another entry with a larger prefix length matches. At the end, the function returns the data from the kept entry and thus the longest matching prefix. For packet delivery, the routing procedure calls the lookup function to get information on how to route the packet. In case of the lookup returns a suitable result, a lower level packet delivery function is called, with the parameters outgoing interface, next-hop and packet. Otherwise, the packet is discarded. So far, neither the destination unreachable message nor a redirect message is implemented. Packets that could not be routed are dropped silently and packets with a better next hop on the same local link are routed without sending a redirect message.

5.1.5 Mobile Router (at home)

As mentioned in the last section, when acting as router, the LIVSIX implementation ignores other router's router advertisements by default. To enable network mobility, the MobileInterface variable is introduced, that is controlled through the Linux sysctl interface. By setting this variable, an administrator can define one interface, on which the router accepts router advertisements, detects movements and auto-configures interface addresses. For mobility management, the existing Mobile IPv6 implementation is reused.

Since, a mobile router behaves as a normal router while it is connected to its home network, this part does not require any additional implementation.

5.1.6 Mobile Router (in a foreign network)

5.1.6.1 Router Advertisements on the Mobile Interface

As soon as a movement to a foreign network is detected, the Mobile IPv6 implementation manages prefix auto-configuration and the binding process with a home agent. Binding updates with correspondent nodes are disabled. Immediately after the movement is detected, an internal variable is set, that indicates, that the mobile router is not attached to its home link. As a consequence, the router advertisement module does not send any router advertisements on the mobile interface any more.

5.1.6.2 Routing Table Adaptation

The only problem that remains is existing entries in the routing table. While at home, all entries are valid, and the default route generally points to a neighbor router in the home network, in a foreign network, all packets sent via routes towards the mobile routers home network must be tunnelled through the mobile router - home agent bi-directional tunnel. In other words, all routes that go by the mobile interface are invalid, and can not be used by the normal packet routing module. On the other hand, as soon as the mobile router returns to its home network, the routes become valid again. To solve this problem, a second routing table, named shadow routing table, is implemented, that holds all routing information that has the mobile interface as destination

interface, while the mobile router is connected to a foreign network. As soon as movement from the home network to any foreign network is detected, a function is invoked, to move all the routing table entries that contain the mobile interface, to the shadow routing table. Vice versa, as soon as a movement from any foreign network to the mobile router’s home network is detected, all entries from the shadow routing table are moved back to the normal routing table.

The word “shadow” in the name of the second routing table is significant for two properties of this table. The first one is of course, that the entries are hidden from the normal routing algorithm. The second one is, that the separation into two routing tables is transparent for normal applications that modify the routing table. “Normal applications” is interpreted as applications that do not support network mobility, e.g. the standard Linux route command (which uses the Linux kernel ioctl interface). Modifications that are caused by such an application are automatically executed on the right table. To achieve the transparent behavior, compared to the normal router, the routing table output is modified:

- “/proc/net/livsix_route” represents the normal routing table,
- “/proc/net/livsix_route_shadow” contains the entries of the shadow routing table
- and “/proc/net/IPv6_route” holds the entries from both routing tables.

Applications, that do not have particular support for LIVSIX network mobility, only use the third table.

5.1.6.3 Packet Routing and Packet Reception

The usage of the shadow routing table assures that the routing function works unmodified at home, and only slightly modified in a foreign network. In a foreign network, first the lookup function on the normal routing table is executed (that contains all entries towards local fixed networks) and only in case this table does not contain a suitable route, the lookup function is called a second time, but now on the shadow routing table. If the second lookup is successful, the packets have to be tunnelled through the mobile router - home agent bi-directional tunnel. The packet tunnelling function of the Mobile IPv6 module is invoked and the packet, that has to be routed, is sent through the bi-directional tunnel, via the access router in the visited network, to the home agent. Packet reception on the mobile router is implemented as a combination of what is done by a router and a mobile host. All packets are processed by the generic packet reception function. For packets that have to be routed, the enhanced routing function, as described above, is executed. Packets that are received through the mobile router - home agent tunnel are unpacked by the mobile host packet decapsulation function, which recalls the packet reception function after decapsulation.

5.1.7 Home Agent for a Mobile Router

Since the functionality of a Mobile IPv6 home agent is already available, the main task during the implementation of the mobile router supporting home agent is the realization of the list of network prefixes that can be reached via the mobile router. For this purpose, the standard routing table is used. For every prefix in a fixed network of a mobile router, the routing table contains one entry with the following fields.

- Destination network address = prefix of the mobile network.
- Destination network prefix length = prefix length of the mobile network.
- Gateway address = home address of the corresponding mobile router.

Upon reception of a packet, the Mobile IPv6 home agent implementation compares the packet destination address to the entries in its binding cache. In case of a matching entry, the packet is tunnelled to the current care-of address of the mobile node. To take advantage of this implementation, the comparison of the destination IP address to the network prefixes that are reachable via a mobile router, has to be done before. That’s why before searching the binding cache, the routing table lookup function is called with the destination IP address as parameter. If the result of the lookup returns a gateway (possibly the IP address of a mobile router), the binding cache is searched for this gateway, otherwise the home agent tries to find a binding cache entry directly for the destination IP address (as a normal home agent does). In both cases, if a care-of address is found, the packet is tunnelled to this care-of address.

Encapsulated packets with a destination IP address that belongs to the home agent are processed the same manner as they are processed by a standard home agent. After decapsulation the home agent delivers them either directly if the destination node is on the same link, or via the home agent’s default router. The above mentioned modifications and additions enable the support of mobile networks. The binding process remains entirely untouched.

5.2 Mobility of IPv6 Multicast Communications

One major objective of the OverDRiVE demonstrator was to experiment the delivery of IPv6 multicast-based services to mobile nodes as well as to moving networks. This section will first present the demonstration story selected as reference for demonstrating multicast for moving networks (including mobile nodes). Then it will summarise the technical approach selected for implementation in the demonstrator. Finally the demonstrator setup and demonstrated features will be detailed.

5.2.1 Demonstration Story

As described in OverDRiVE deliverable D03 [1], there are many use cases highlighting the need for delivery of IPv6 multicast to mobile nodes as well as to fixed or mobile nodes in moving networks. Typical examples include:

- Device/Vehicle Management Operations:
 - An operator (or device manufacturer) can take advantage of IPv6 multicast for mass software delivery/upgrade to a group of devices. The software can be of any type, e.g. a dedicated application reserved for a given set of customers, an update of the devices configuration, a patch in order to correct a bug of the phones’ firmware, etc.
 - Similarly, a car manufacturer can use IPv6 multicast to ease mass software delivery/upgrade to a fleet of vehicles.
- Support of group communication applications for end users. For instance, a vehicle’s passengers participating in group communications, such as audio/video streaming or conferencing.
- Etc.

The demonstration story selected for demonstrating multicast for moving networks covers the second use case as mentioned above. Here is a description, copied from [1]:

“Bob is just going out of the plane, back from his 2-week business trip in Japan. While walking in the WLAN-equipped airport towards the luggage belt, Bob switches on his personal device to

watch the live evening news. While viewing the summary of the last match of his favorite football team, Bob managed to get his suitcase and has walked to the entrance of the airport to catch a Taxi. When entering the Taxi, Bob’s session is handed-over to the intra-vehicular WLAN network and is routed through the UMTS access supported by the Taxi. While the Taxi is on the way to Bob’s home, his communication is then automatically handed-over from UMTS to a DVB-T cell, by the network operator, due to the high popularity of the content in this area.”

Of course, the technology implemented in the demonstrator to support delivery of IPv6 multicast to mobile nodes, possibly visiting moving networks, is generic and can be use irrespective to the use case to be supported.

Technical Approach

Various approaches have been identified in OverDRiVE to enable delivery of IP multicast to mobile nodes and moving networks (see OverDRiVE deliverables D04 [2], D09 [4]). Among all of them, the *remote subscription* approach has been selected as the base for the OverDRiVE’s mobile multicast architecture and demonstrator. The reason for this choice is discussed in details in deliverables D09 [4] and D16 [5], but can be summarized by saying that the remote subscription is the best candidate to meet OverDRiVE’s mobile multicast requirements (see D03 [1]). Indeed the approach natively offers key capabilities such as:

- Multicast receivers’ mobility,
- Per-flow handover,
- Preservation of multicast nature of the traffic all along the routing path, as a way to optimize network and radio resources,
- Multicast traffic routing along the optimal path (no “triangular routing”).

In addition, protocol extensions have been designed in OverDRiVE to support seamless mobility of multicast receivers (see deliverable D09 [4] and D16 [5]).

In OverDRiVE, remote subscription is used both for mobile hosts and mobile routers (serving moving networks). Indeed, similarly to a mobile multicast receiver, the mobile router maintains ongoing IPv6 multicast sessions by joining the multicast group, with the MLD [7] protocol, through the local multicast router each time it changes IPv6 subnet (remote subscription).

A key difference however is that the mobile router (MR), as opposed to the mobile host/receiver, does not have group membership information locally available. This is because multicast applications (e.g. video streaming) are not run on the mobile router itself, but instead on the nodes behind MR in the moving network. Thus, there is a need for MR to collect those group membership informations (i.e. multicast groups of interest for nodes in the moving network) and subscribe to those groups on behalf of the moving network nodes in order to receive the related traffic. A second important requirement for the mobile router is to allow optimal routing of multicast traffic in the moving network, thus routing multicast packets only in parts of the moving network where receivers are located. *MLD-based Multicast Forwarding* [8] has been identified as a candidate for deployment within the moving network in order to solve the above issues. Indeed, the use of MLD-based Multicast Forwarding within the moving network allows MR to collect group membership information from nodes within the vehicle, as well as optimal routing of multicast packets within the moving network. Details of how MLD-based Multicast Forwarding is used in the context of moving networks are presented in D09 [4] and D16 [5].

This approach is well-suited for vehicular environments (small to medium networks) and exhibits the following advantages:

- Enable global mobility in the IPv6 multicast Internet (use of MLD protocol at MR).

- Optimal routing, even with nested mobile routers.
- Per-flow handovers are possible, for MR equipped with multiple egress interfaces.
- Compatibility with seamless mobile multicast mechanisms.
- Independent of the base NEMO support [9]
- No need to run a multicast routing protocol within the moving network.
- Easy to implement.

5.2.2 Demonstrator Configuration

The overall objective of the demonstration is to show uninterrupted delivery of IPv6 multicast video streaming to a mobile multicast receiver when:

- Changing its attachment point to the infrastructure,
- Entering an IPv6 moving network (such as a car),
- Located within a moving network, whose Mobile Router (MR) is changing its own attachment point to the infrastructure.

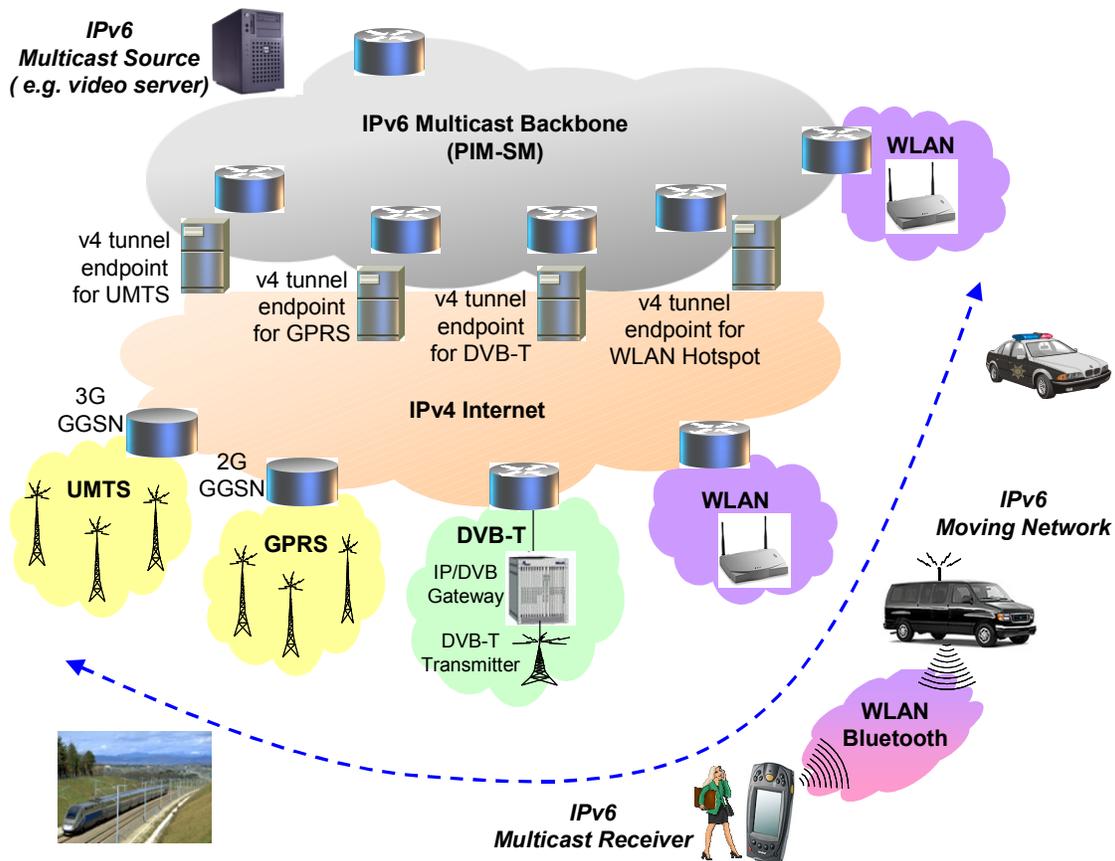


Figure 21: IPv6 multicast for moving networks – Full demonstrator architecture

The overall demonstrator architecture is depicted in Figure 21. The network topology is made of an IPv6 backbone running the PIM-SM [6] multicast routing protocol, and interconnecting heterogeneous access networks including 802.11b WLAN, 2G cellular GPRS, 3G cellular UMTS,

and broadcast DVB-T, between which the mobile multicast receiver and the moving network can roam. The moving network is also offering Ethernet and 802.11b WLAN (and possibly Bluetooth) connectivity for its local nodes.

Routers in the infrastructure are running the FreeBSD operating system offering PIM-SM support for IPv6 thanks to the pim6sd daemon [10]. The following pim6sd.conf configuration files can be used for:

- One or several candidate Rendezvous-Points (RP):

```
cand_rp;  
cand_bootstrap_router;  
log;
```

- All Designated Routers (DR):

```
log;
```

Because existing GPRS and UMTS services commercially available do not offer support for IPv6, IPv4 tunneling is used to convey IPv6 multicast over those 2G and 3G cellular systems in a transparent manner. A simple IPv6-in-IPv4 encapsulation can be used when a public IPv4 address can be obtained from the cellular system, but most of the time IPv6-in-UDP/IPv4 encapsulation is required in order to traverse Network Address Translators (NAT) deployed at the edge of the cellular system. Similarly, because most of existing DVB-T/IP gateway products currently available on the market do not support yet transport of IPv6 packets over DVB-T (although this is allowed by the MPE standard, see [2] section 2.3.2), IPv6-in-IPv4 encapsulation is required to allow delivery of IPv6 multicast packets to DVB-T receivers. As illustrated in Figure 21, dedicated tunnel endpoints are deployed in the demonstrator to relay IPv6 multicast packets between the IPv6 backbone and the mobile node (MN) or mobile router (MR) through an IPv4 tunnel. Similarly, corresponding tunnelling/detunnelling function is needed on MN/MR side. Although this function could be co-located on the MN/MR entity itself, it has been placed on a separate physical entity (called GPRS-, UMTS- or DVB-T-Front-Boxes, see section 7) directly attached to the MN/MR. The main reason for placing the “mobility management” function and “tunneling function” on different physical entities was to ease implementation by different partners and minimize risk during integration.

The IPv6 multicast source used in the demonstrator is a video streaming source running the VLC media player version 0.6.2 [11] placed on a Linux Desktop PC.

The mobile multicast receivers and the mobile router can attach to the IPv6 multicast backbone through GPRS, UMTS or DVB-T thanks to their own Front-Boxes playing the role of interfaces towards the respective networks, or directly through WLAN thanks to a local 802.11b WLAN network interface card.

The mobile router, the mobile multicast receivers, the fixed multicast receivers placed in the moving network, and the fixed routers in the moving network are laptop PCs running Linux 2.4.x kernel extended with LIVSIX Motorola’s open source IPv6 stack [12].

MLD protocol [7] and MLD-based Multicast Forwarding [8] support have been implemented in LIVSIX. Details about the implementation and validation are presented in OverDRiVE deliverable D16 [5].

The following MLD and MLD-proxy configurations are used:

- The **mobile multicast receivers** (as well as fixed receivers in the moving network) have the “host part” of the MLD protocol enabled on all of their network interfaces. This configuration comes by default with LIVSIX. One can also use the /proc file system to activate or deactivate this configuration:
 - Activation of MLD “host part” on network interface nic0:
echo 1 > /proc/sys/net/livsix/conf/nic0/mld_listener
 - De-activation of MLD “host part” on network interface nic0:
echo 0 > /proc/sys/net/livsix/conf/nic0/mld_listener
- All **fixed routers in the moving network** are configured with one upstream interface running the “host part” of MLD protocol, while all the other interfaces are configured as downstream interfaces and run the “router part” of MLD. The upstream interface of a fixed router should be chosen as the local interface towards the MR, which is the root of the MLD-based Multicast Forwarding tree. In addition, in order to route IPv6 multicast packets, all those fixed routers are configured as “MLD proxy”. Again, one can use the /proc file system to activate this configuration. For instance on a fixed router equipped with 3 interfaces (nic0, nic1 and nic2) the following configuration can be used to have nic0 as upstream interface and nic1 and nic2 as downstream interfaces:
echo 1 > /proc/sys/net/livsix/conf/nic0/mld_listener
echo 0 > /proc/sys/net/livsix/conf/nic0/mld_router
echo 0 > /proc/sys/net/livsix/conf/nic1/mld_listener
echo 1 > /proc/sys/net/livsix/conf/nic1/mld_router
echo 0 > /proc/sys/net/livsix/conf/nic2/mld_listener
echo 1 > /proc/sys/net/livsix/conf/nic2/mld_router
echo “nic0” > /proc/sys/net/livsix/mld_proxy_default_ifc
- The **mobile router** has the “host part” of the MLD protocol enabled on all of its egress network interfaces, while all its ingress interfaces (i.e. towards the internal of the moving network) are running the “router part” of MLD.

5.2.3 HyWIN 2003 Demonstrator

A simplified version of the demonstrator has been shown during the HyWIN 2003 workshop, without limiting in any way the applicability of the technology demonstrated.

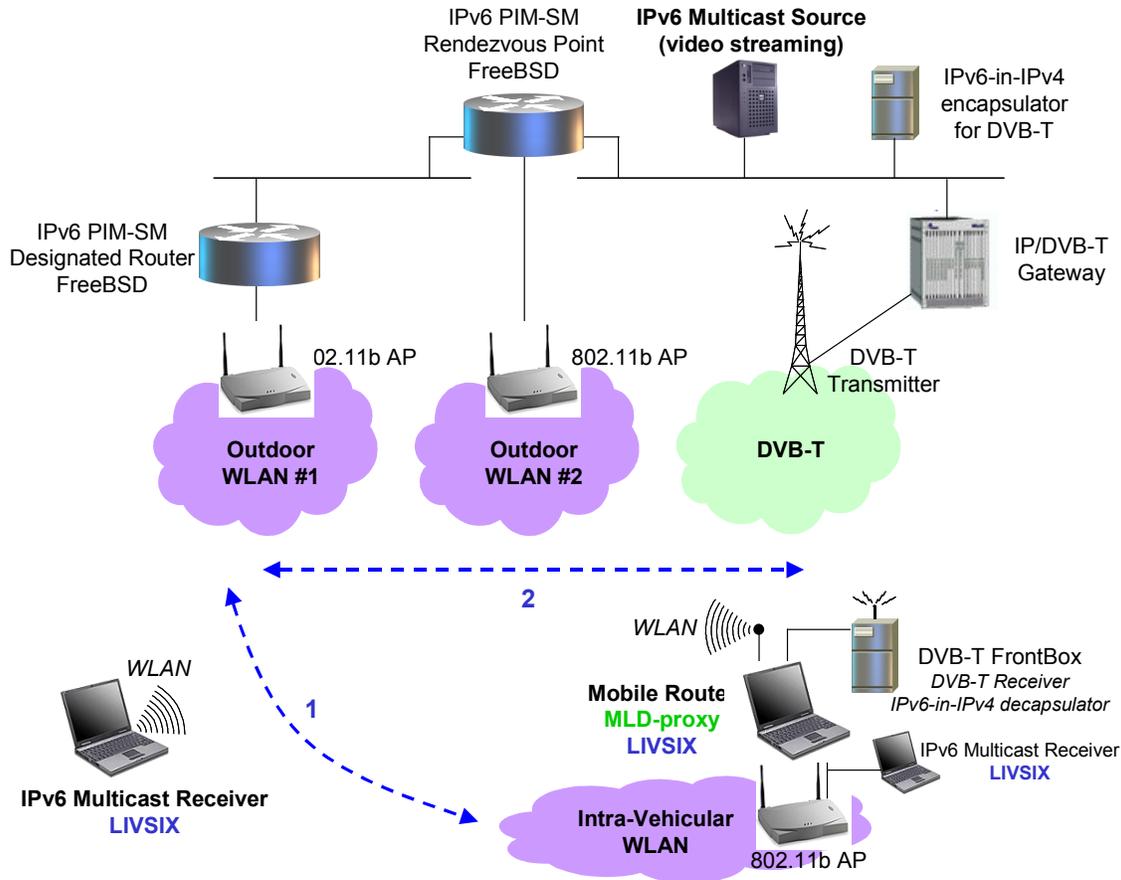


Figure 22: IPv6 multicast for moving networks – HyWIN 2003 demonstrator

The network topology, illustrated in Figure 22, is made of an IPv6 backbone including two PIM-SM multicast router, and interconnecting three access IPv6 subnets: two 802.11b WLAN IPv6 subnets, and one DVB-T IPv6 subnet. The PIM-SM routers are configured as described in section 5.2.2.

The IPv6 multicast source used in the demonstrator is a video streaming source running the VLC media player version 0.6.2 [11] placed on a Linux Desktop PC.

The mobile router can attach to the IPv6 multicast backbone through DVB-T thanks to the DVB-T Front-Box equipped with the DVB-T RX card and implementing the IPv6-in-IPv4 de-tunneling function, or directly through Outdoor WLANs #1 or #2 thanks to its 802.11b network interface card. The internal link of the moving network is made of a single IPv6 subnet supporting both Ethernet and 802.11b technologies. A fixed host in the moving network connected via Ethernet will be used as a multicast receiver for the video streaming session.

In addition, two mobile multicast receivers are configured in the demonstrator. Both of them are equipped with a single 802.11b network interface card. The first one always stays attached to the same IPv6 subnet (Outdoor WLAN#1) and serves as “witness receiver”, while the second one moves in the topology according to the following mobility pattern:

1. Starting receiving multicast video streaming through Outdoor WLAN #1
2. Moving into the Intra-vehicular WLAN by attaching to the embedded access point, and maintaining the multicast video streaming thanks to the mobile router (running MLD-proxy) itself connected at Outdoor WLAN #1

3. While the multicast receivers in the moving network (the visiting one connected via WLAN, and the fixed one connected via Ethernet) are receiving the video streaming, MR is moving to Outdoor WLAN #2
4. While the multicast receivers in the moving network are receiving the video streaming, MR is moving to DVB-T

The following mobility scenario demonstrates how the multicast session to a mobile receiver is maintained as the node is moving within the infrastructure, moving into a moving network, and when the moving network itself is moving.

The mobile router, the mobile multicast receivers and the fixed multicast receiver placed in the moving network are laptop PCs running Linux 2.4.x kernel extended with LIVSIX Motorola’s open source IPv6 stack [12].

As discussed in section 5.2.2, the mobile multicast receivers (and the fixed receiver in the moving network) have the “host part” of the MLD protocol enabled on their 802.11b network interface. The mobile router has the “host part” of the MLD protocol enabled on its 802.11b egress network interface as well as on its Ethernet interface towards the DVB-T FrontBox. The MR’s unique ingress interface (Ethernet) is running the “router part” of MLD.

The demonstrator features Graphical User Interfaces (GUIs) both on the mobile multicast receiver and the mobile router displaying the path followed by the IPv6 multicast packets as the mobile receiver (MN) or the mobile router (MR) are moving.

Figure 23 shows the GUIs at startup, before the MN starts the video streaming client application. The buttons on the top of the GUIs allow to select the network to switch multicast traffic to. The pictures at the bottom of the GUIs show the various networks available. No multicast traffic is shown because the application is not started yet.

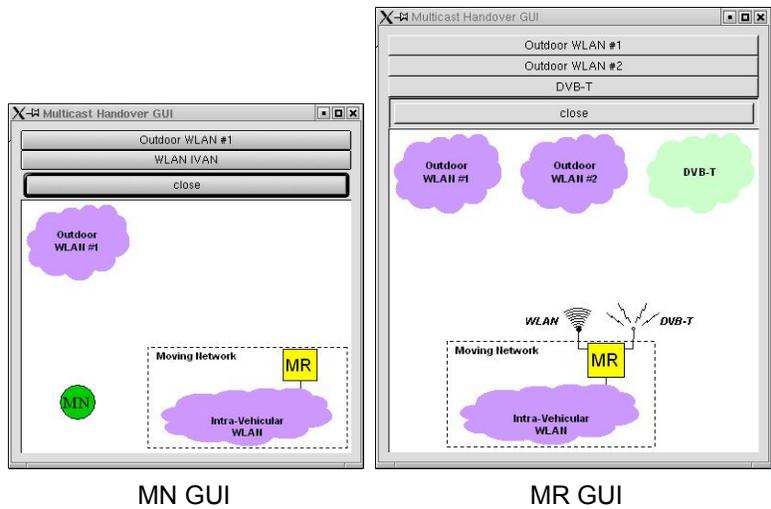


Figure 23: MN and MR Multicast Handover GUIs – Startup.

Figure 24 shows the GUIs once the video streaming client application has been started on MN (step 1). The subscription to the multicast group has been handled through Outdoor WLAN #1 access and IPv6 multicast packets are now routed towards MN. This is highlighted by the “orange” arrow on the MN’s GUI. At this time the video streaming is displayed on the MN’s screen. Note that no multicast traffic is routed through the moving network up to now.

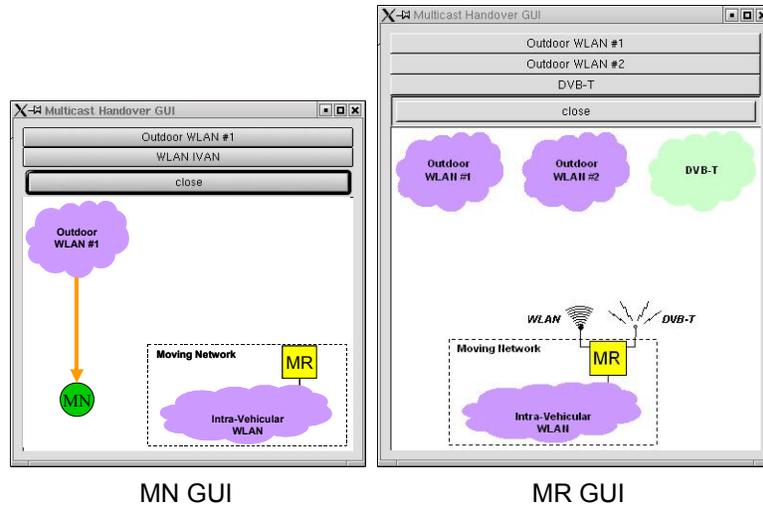


Figure 24: MN and MR Multicast Handover GUIs – MN starts video streaming client.

Figure 25 shows the GUIs at step 2, when the MN handovers his video streaming to the Intra-Vehicular WLAN by attaching to the embedded access point. This action is performed by clicking on the “WLAN IVAN” button of MN’s GUI. As soon as the handover is triggered MN re-subscribes to the multicast group from within the moving network. MR detects that a multicast receiver for this group is now attached to the internal WLAN access and thus subscribes to the group through Outdoor WLAN #1. At this point, multicast traffic is now received by MR and relayed to MN through the Intra-Vehicular WLAN. The multicast video streaming is maintained, and continues displaying on the MN’s screen.

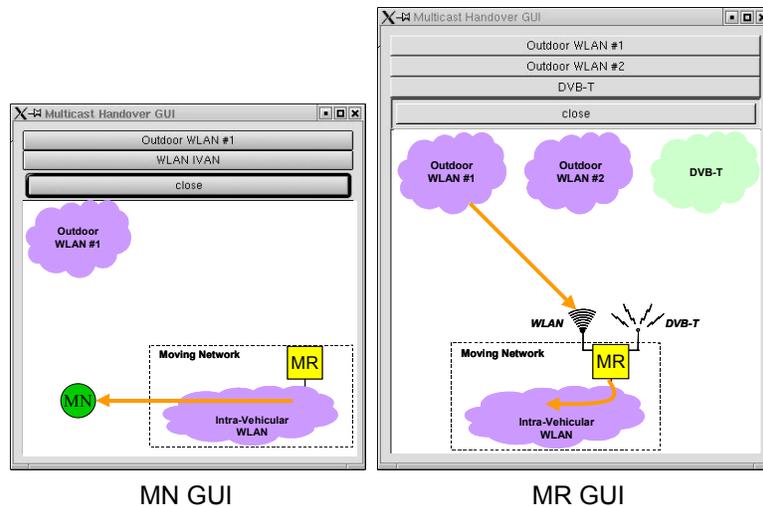


Figure 25: MN and MR Multicast Handover GUIs – MN moves into the moving network.

Figure 26 shows the GUIs at step 3, when MR is moving from Outdoor WLAN #1 to Outdoor WLAN #2. This action is performed by clicking on the corresponding button of MR’s GUI. As soon as the handover is triggered MR re-subscribes to the multicast group through the new WLAN access, receives multicast traffic, and continues to relay it to MN through the Intra-Vehicular WLAN. Again, the multicast video streaming is maintained.

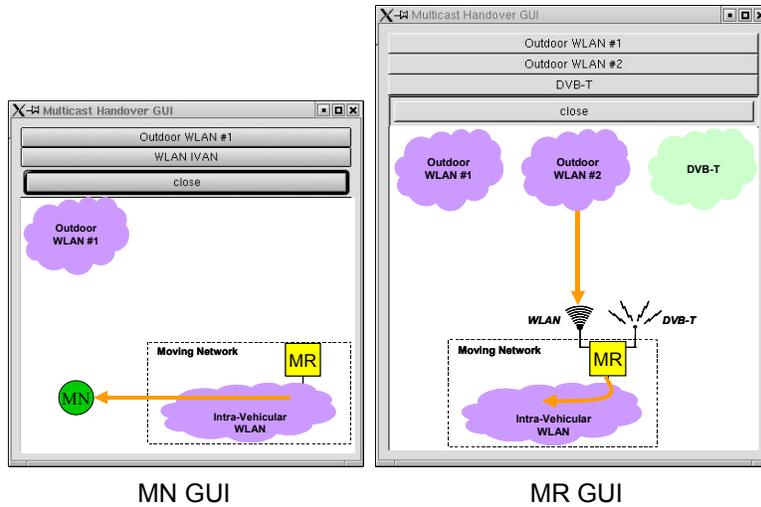


Figure 26: MN and MR Multicast Handover GUIs – MR moves to Outdoor WLAN #2.

Figure 27 shows the GUIs at step 3, when MR is moving from Outdoor WLAN #2 to DVB-T. This action is performed by clicking on the “DVB-T” button of MR’s GUI. As soon as the handover is complete, multicast traffic is now received on the MR’s interface connected to the DVB-T FrontBox, and relayed by MR within the moving network. Again, the multicast video streaming is maintained.

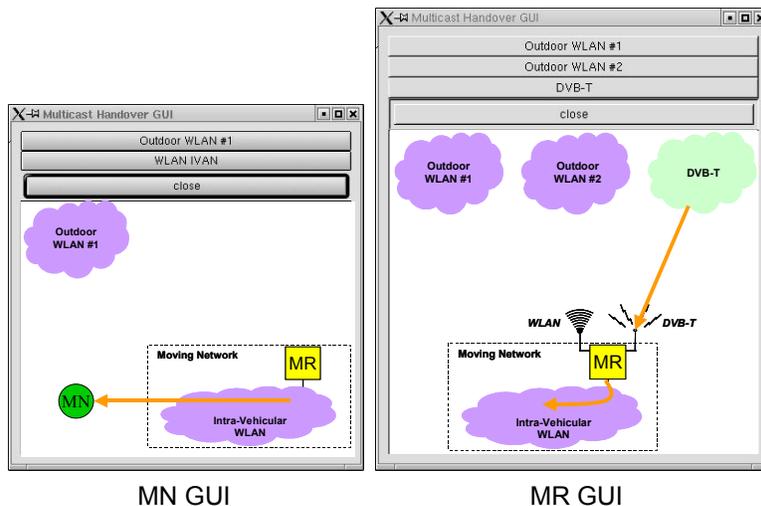


Figure 27: MN and MR Multicast Handover GUIs – MR moves to DVB-T.

Figure 28 shows the poster of the “Multicast for Moving Network” demonstration given at the HyWIN 2003 workshop, in Turin, Italy, on December 2nd 2003.

5.3 Seamless IPv6 Multicast Handovers

The seamless multicast mobility demonstration aimed at presenting seamless multicast mobility in the OverDRiVE environment. The provisioning of flow-level mobility raises several issues, as stated in D04. The present demonstration focused only on the details related to mobility management. In particular, it showed that, if the enhancements proposed during the project are implemented, then the Remote Subscription approach can satisfy the requirements of the OverDRiVE project. The main feature of the proposed solution is that it maintains the optimal structure of the multicast tree, and the possibility of per-flow handovers, while it assures seamless handovers among access points. These features should work even in a hybrid environment, i.e., where multiple access technologies are present.

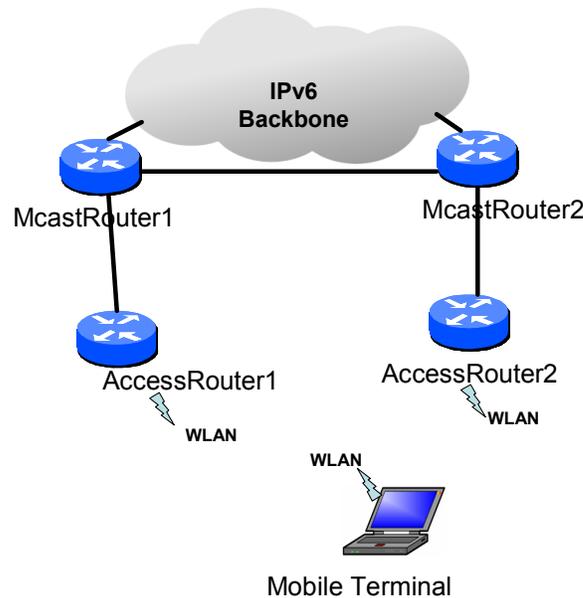


Figure 29: Multicast Routers in the Access Network

In our model, the access network contains a mesh of multicast routers (MR) that, together with the IPv6 backbone, provide the multicast tree (see Figure 29). The MRs are standard IPv6 multicast routers without any modifications. At the wireless edge of the access network there are several access routers (AR) that do not have multicast capabilities. They are responsible of the mobility management of the mobile routers and terminals. In order to assure multicast capabilities to the mobile entities, the access routers work as multicast signalling (Multicast Listener Discovery - MLD) proxies.

Mobile Terminals (MT) are always logged on exactly one AR. This implies that the AR maintains the context (i.e., AAA, address, mobility related information) of each logged MT. The wireless interfaces of the ARs accept packets only from the logged MTs. Within OverDRiVE, multiple access technologies are present. To provide a flexible support to these different access modes in testbeds and prototypes, OverDRiVE introduced a modular system of virtual interfaces. A MT that has to support a new technology (e.g., GPRS) uses a common Fast Ethernet interface for the access. That interface is connected to the virtual interface module, which is referred to as FrontBox in OverDRiVE. The FrontBox is a dedicated router that has a Fast Ethernet interface towards the MT and the required wireless interface that assures the connection to the AR.

Not only Mobile Terminals, but also entire Moving Networks can connect to ARs. Moving networks are composed of a Mobile Router and one or more Mobile Terminals. Mobile Routers act as proxies between ARs and MTs. MRs have both Mobile Terminal and Access Router functionalities: Mobile Routers log in to ARs as hosts do, and are legacy Mobile Terminals from the ARs’ point of view. Besides, they also act as Access Routers, and are seen by Mobile Terminals as legacy ARs.

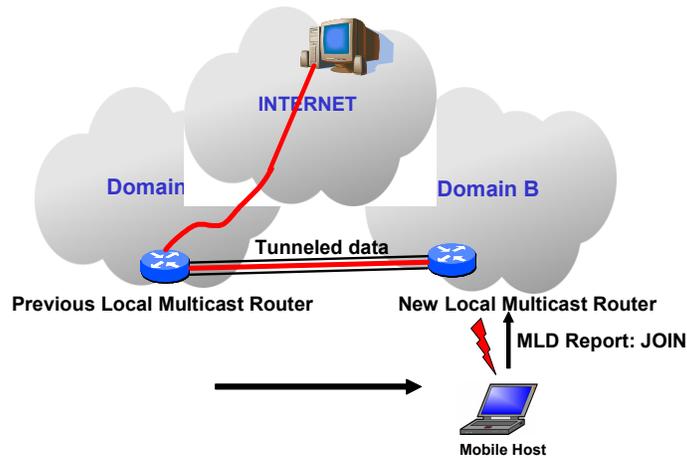


Figure 30: Seamless Multicast Handover Solution

A seamless multicast handover with Remote Subscription means that Mobile Terminals subscribe to multicast groups at the AR they are currently logged in to. Therefore, the multicast tree must be rebuilt every time the MT changes its point of attachment. Rebuilding the multicast tree may be time consuming, and this may result in packet loss. To minimize packet losses and to ensure seamless handover, ARs keep track of their Mobile Terminals (context management); when a MT wishes to handover to a New AR, the Previous AR tunnels multicast data to the New AR if necessary, using IPv6 in IPv6 encapsulation, until the multicast tree is rebuilt (see Figure 30).

5.3.1 Demonstration

The demonstration scenario tries to highlight the above stated features. The testbed network used Open Source Operating Systems based PCs as routers. The IPv6 multicast routers used FreeBSD and PIM-SM routing. The access routers and the mobile terminals used Linux Debian 3.0 OS.

The applications used for the demonstrations were two Linux based media streaming tools. The sender PC used *VideoLAN (vlc)* to generate the multicast stream. It generated a 700kbit/sec stream, which can be transmitted over WLAN. However, GPRS can not support such speeds. In order to demonstrate the smooth WLAN-to-GPRS handover, we used a down-coded version of the same streaming content, at the speed of 20kbit/sec. In this latter case, the application we used was *vic*. Note that *vlc* transferred both video and voice, while *vic* only video..

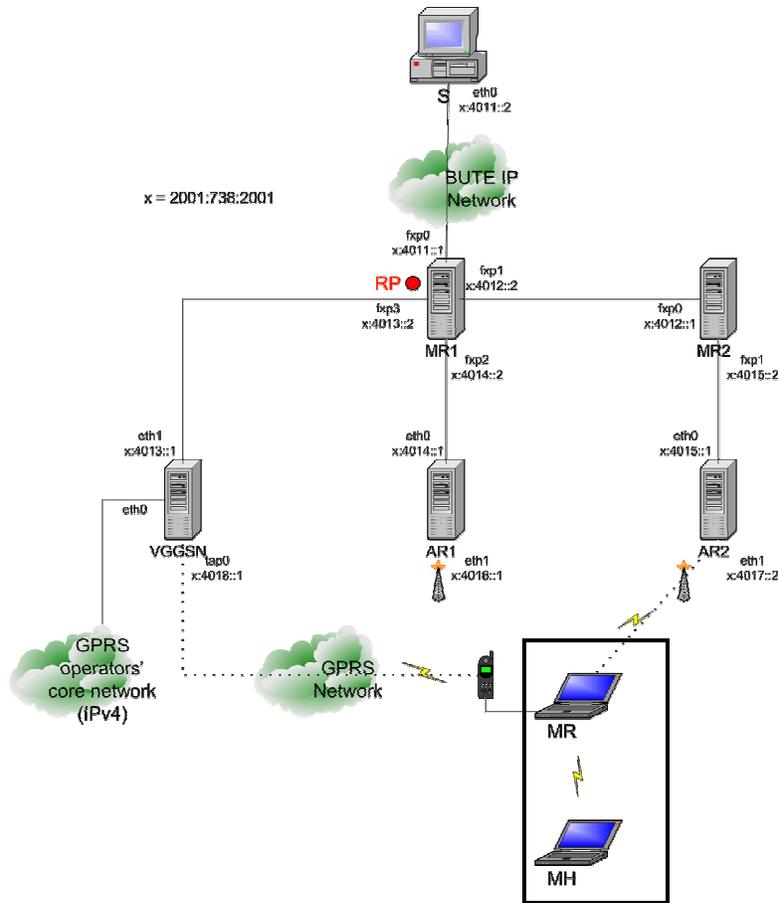


Figure 31: The topology of the testbed

The topology of the testbed used for demonstrations is presented in Figure 31. The ARs implement the enhanced tunnelling mechanisms to support seamless multicast handovers. MTs also have to support the modified multicast mobility management scheme.

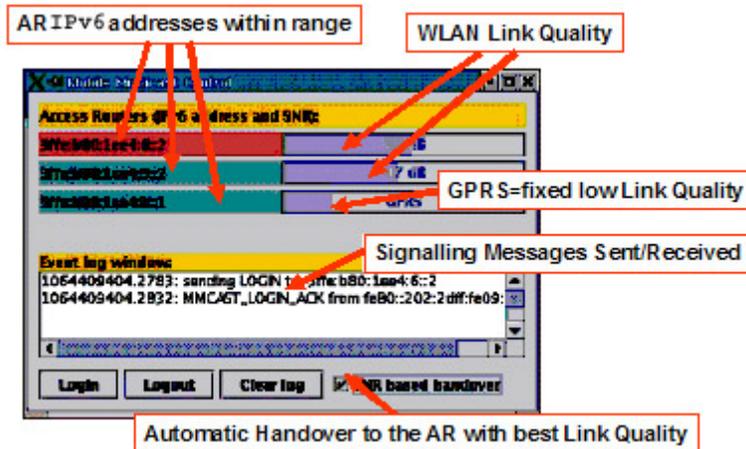


Figure 32: The GUI for multicast handover control

The Graphical User Interface (GUI) used to control and monitor the handovers is presented in Figure 32.. Each AR within range (including GPRS virtual access, if GPRS connection is available) is represented with a horizontal bar, specifying the IPv6 address of the AR, and the

sensed signal-to-noise (S/N) ratio. We assigned a very low S/N ratio to the GPRS connection, one that is always worse than the S/N ratio of an acceptable WLAN connection. The GUI can force an automatic mode, where the MT always chooses the AR with the best signal-to-noise ratio. Alternatively, the MT can work in manual mode, when the user has to click on the desired AR.

5.3.2 Handovers

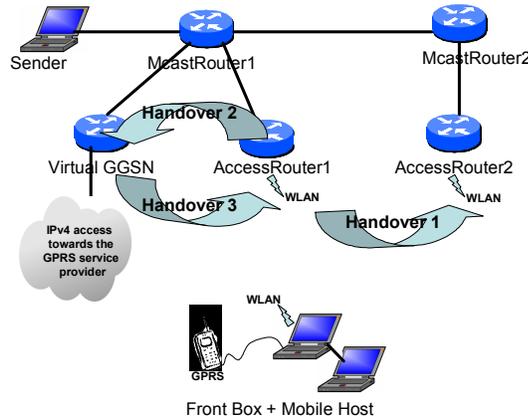


Figure 33: Handover types for the Mobile Terminal

The demonstration aimed at validating the seamless multicast handover concept through the successful realization of several handover types. For this purpose, we have selected three handover types, while the Mobile Terminal is directly connected to the access routers, as shown in Figure 33:

Handover 1: WLAN-to-WLAN

Handover 2: WLAN-to-GPRS

Handover 3: GPRS-to-WLAN

Further on, we also tested and demonstrated the possibility of the IVAN support. Our test focused only on the mobility issues, namely on moving into and out of the IVAN. The issues regarding the multicast tree management inside the IVAN are discussed in section 7. In this case, the MT used only its WLAN interface:

Handover 4: WLAN-to-WLAN (moving into the IVAN)

Handover 5: WLAN-to-WLAN (leaving the IVAN)

While the MT is in the IVAN (it is 'bound' to MR), the Mobile Router may handover as well. In this situation we have the following possible cases:

Handover 6: WLAN-to-WLAN (MR handover)

Handover 7: WLAN-to-GPRS (MR handover)

Handover 8: GPRS-to-WLAN (MR handover)

6 Group Management for Mobile Multicast

The objective of the Group Management is to group the members of a multicast group

- within UTRAN, choosing between several point-to-point (ptp) transmissions or one point-to-multipoint (ptm) transmission, but without power control and thus less efficient than one power controlled ptp transmission,
- within a hierarchical system, selecting the most efficient cell hierarchy level,
- according to terminal capabilities.

In the figure below the lab setup is depicted. Parts of the Group Management are in the clients and parts in the network. Each WLAN access point represents a UMTS cell and the access router has a traffic shaper to reduce the bandwidth accordingly. The Group Management in the network is centralised in one node, with a database, whereas in a real UMTS system it would be distributed within the core and radio network.

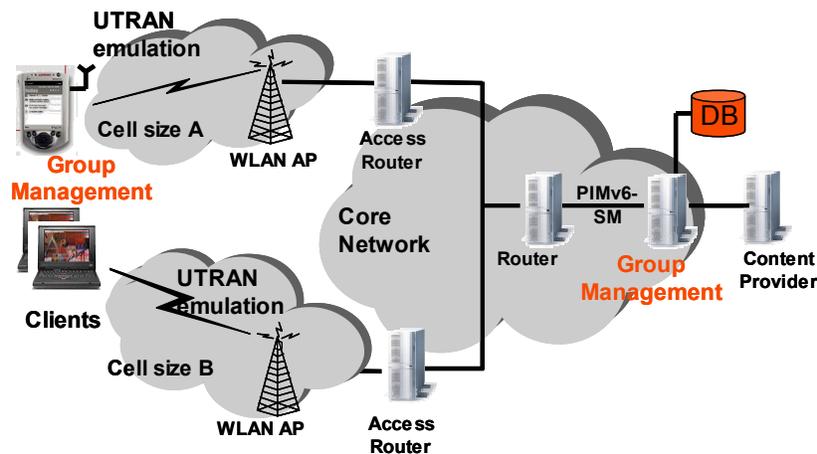


Figure 34: Group Management lab setup

In the network PIMv6-SM is used as a multicast routing protocol. As a service, short clips (MMS like) from a football game are provided. The IPv6 content server sends the message once to a multicast address. All registered users/clients know already to which multicast address they have to listen to and get the content more or less at the same time.

Up to now this is the current existing method to deliver the same multimedia content to a large group of mobile users. With the introduction of the Group Management we aim at optimising this delivery and, if possible, the user satisfaction as well, by managing the group of interested users and influence the technology used to deliver this content by, for instance, choosing between using ptp (unicast) or ptm (multicast) connections. The network controls one part and the other one is executed at the terminal side. Both processes should interact with each other and exchange information.

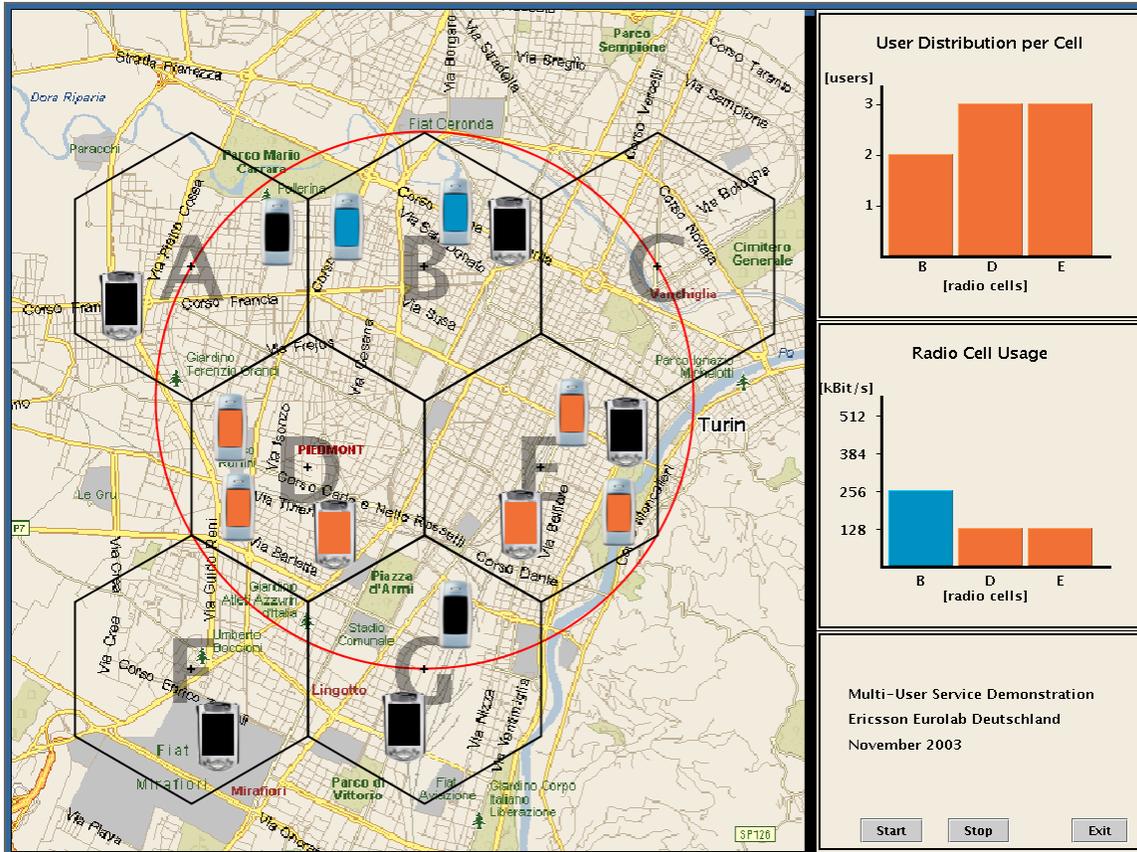


Figure 35: Screen-Shot of the Group Management GUI

The Graphical User Interface (GUI) of the demonstration shall show the benefits and gains out of mobile multicast approach. It shows the network topology and the actual member distribution in a certain area. In Figure 35: Screen-Shot of the Group Management GUI, an example network topology with a two cell layers is depicted. The mobile terminals are distributed and associated to certain base stations. The number of mobile terminals per base station is shown top right in a separate graph. The group partitioning uses these figures to determine whether a point-to-point radio bearer (dedication communication resources) or a point-to-multipoint radio bearer (common communication resources) shall be used. The graph below, the “Radio Cell Usage” is depicted. In this example, the cells D and E are serving a 128kbps flow using point-to-multipoint radio channels and the cell B serves two Ues with each a 128kbps (= 256kbps) transmission using a point-to-point radio channel. Note, that we have chosen purpose a rather low threshold of switching between point-to-point and point-to-multipoint channels for demonstration. In reality, this threshold is likely in the area of 7 Ues per cell.

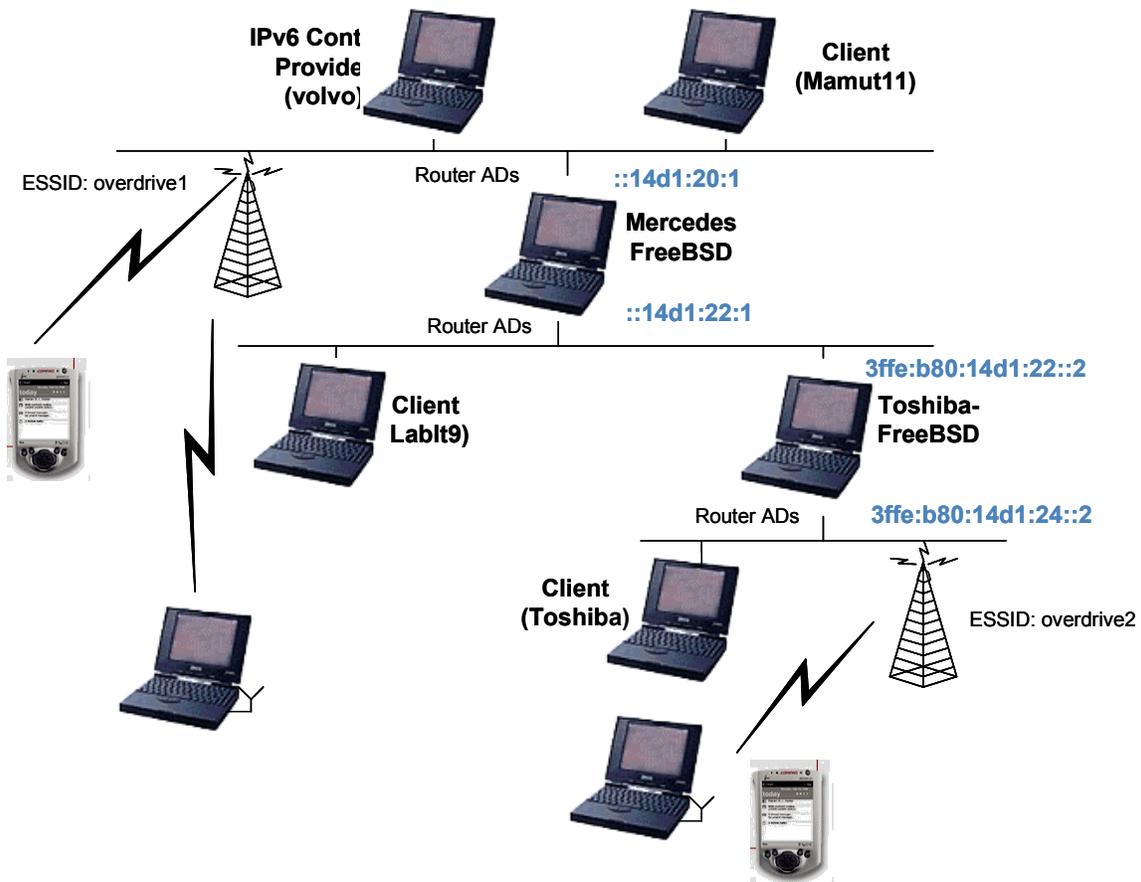


Figure 36: Topology of the Group Management Demonstration

The set-up of the Group Management demonstration is depicted in Figure 36. The Server part of the Group Management application is executed on the IPv6 Content Provider (Volvo). The Clients are connected via up to two multicast routers to the content server. Two WLAN access point provide WLAN connectivity for mobile clients. One Demonstration client moved (one mouse-click) from one WLAN access point to the other and changed by the the point of attachment in the topology.

Note, for time reasons, it was not possible to integrate the Group Management demonstration in the overall demonstration set-up.

7 Demonstrator Services

7.1 Introduction

The demonstrator story encompasses the following eight services:

1. Video Streaming:
Demonstration of a video transmission from a server outside the vehicle to a mobile device connected to the vehicle's IVAN.
2. Web Broadcast:
Web Broadcast over DVB-T allows to send Web pages over a DVB-T cell, with both general and location based content.
3. Multicast Messaging and Streaming:
Demonstration of video streaming to a multicast group, including messaging.
4. Adaptive Video on Demand:
Demonstration of a service that can be used on a roaming mobile device inside and outside the IVAN and adapt to different link characteristics.
5. Remote Access:
Remote control of car functions and remote access of a web server located inside the car.
6. Software Download:
Downloading of a new software release to increase the functionality of the in-vehicle IVAN or the functionality of the vehicle.
7. Web Access:
Dynamic access to Web pages from inside the car.

In the following sections the services are described in more detail.

7.2 Video Streaming

We used video streaming to demonstrate the multicast features of our demonstrator.

7.2.1 Realization

The video transmission is realized with VideoLAN (<http://www.videolan.org>) and video conferencing is realized with the multicast tools VIC/VAT. The video server is located in the IPv6 network. The VideoLAN transmission can use any of the access systems available in the testbed: WLAN, UMTS, GPRS, and DVB-T.

The VideoLAN VLC media player [11] version 0.6.2 (or above) has been used for the demonstration, both on the server and client sides. VLC is a highly portable multimedia player for various audio and video formats (MPEG-1, MPEG-2, MPEG-4, DivX, mp3, ogg, ...) as well as DVDs, VCDs, and various streaming protocols. It can also be used to stream in unicast or multicast in IPv4 or IPv6. In the demonstration, VLC has been used with RTP/UDP transport over IPv6 multicast.

7.2.2 IVAN Set Up

This section describes the services available mainly from the DVB-T multiplex; interactions with the 3G network to request special information will be also described. The DVB-T Front-Box inside the car provided the car with the services described below.

Standard Audio/Video DVB-T programs

The DVB-T multiplex allowed to provide the IVAN with generic contents. The contents can be divided in standard MPEG-2 television contents (RaiUno and OverDRiVE Turin) and IP Contents (MPEG-4 Audio/Video Streaming over IPv6 and Web Pages over IPv6 BTFTP).

Audio/Video MPEG-2 standard DVB-T programs are received by the car using the DVB-T Front-Box. However this kind of contents could be displayed using any kind of DVB-T compliant Set-Top-Box. For the Demonstration, the DVB-T bouquet for OverDRiVE included:

- One localized MPEG-2 program, describing the city of Turin (place of the Demonstration).
- One live MPEG-2 program from the Rai offering (RaiUno).

These contents, for the Demonstration, have been displayed on the screen of the Front-Box itself. The video output could be easily switched to the main screen of the car.

Streaming Audio/Video Contents

The IPv6 data included in the DVB-T transport stream comprised also MPEG-4 low bit-rate video streaming packets, with localized contents (in this case, a video about the city of Turin, see Figure 37), to be used for the access through WLAN of Handheld devices.

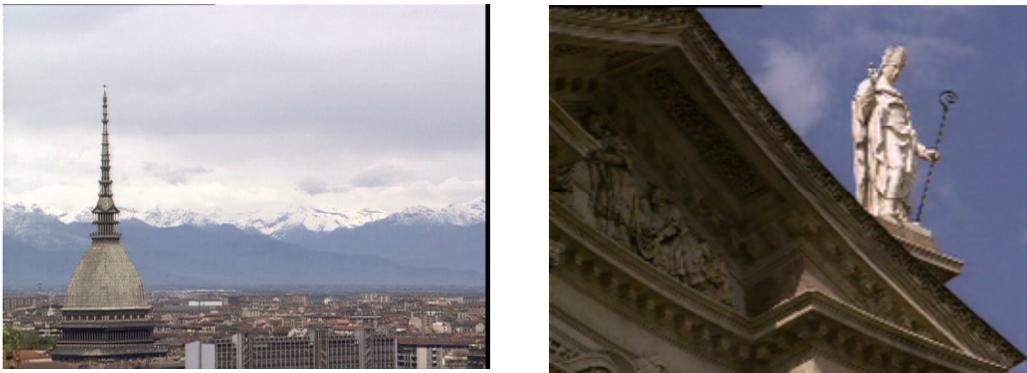


Figure 37: DVB-T Video Screenshots

The Operating System of the Front-Box was Linux. The choice of Linux was due to the requirements of the Front-Box:

- Support of the latest drivers of the TechnoTrend DVB-T reception card
- Support of IPv6
- Availability of open networking software and development tools

The Redhat 9.0 Linux version was chosen. The DVB-T reception card, TechnoTrend DVB-T card with an integrated MPEG-2 decoder, has been easily set up for this platform. The Linux drivers [<http://www.linuxtv.org>] provided the possibility of IP data extraction and real-time hardware MPEG-2 decoding and overlay on the PC screen.

For a detailed description of the Front-Box, IPv6 over IPv4 insertion and extraction, see section 8.2.

7.2.3 Content Server

For each type of service provided to the IVAN, a Content Server was used. In the following the Content Server Hardware and Software characteristics will be described, as well as the Interfaces between them and the overall Demonstrator Platform. This section describes the Video Streaming setup.

Standard Audio/Video Content Server

For the Demonstrator two kinds of video contents over DVB-T were used. The first one, was the live RaiUno video, re-multiplexed in the OverDRiVE bouquet; the source of this content is the live satellite RaiUno channel.

The second one was the ad hoc OverDRiVE Torino video clip, which was specially included for the Demo. The video clip was recorded as MPEG-2 transport stream file and it was played in loop with a software video player and sent in the MPEG-2 transport stream. The equipment to do that was a Thales Opal Gateway.

Streaming Audio/Video Contents

The video clip used for the Demonstration MPEG-4 video streaming was stored on a Streaming Server, based on a RedHat 9.0 Linux PC. The streaming software was VideoLAN vlc 0.6.2, downloaded from www.videolan.org that was setup choosing a multicast IPv6 address for the streaming.

7.2.4 Content

Table 1 summarizes the contents used for the Demonstration in the car. The content will be described using its type, format and specifying the possible transport protocols for it (the reason is that the features of the contents imply sometimes one specific transport technology).

| Content Type | Format | Transport | Short Description |
|-----------------------------|-------------------|---|--|
| RaiUno Audio/Video Live | MPEG-2 | MPEG-2 Transport Stream, DVB-T | Live channel broadcasted by Rai |
| OverDRiVE Turin Audio/Video | MPEG-2 MPEG-4 | MPEG-2 Transport Stream, DVB-T IPv6 over DVB MPE, DVB-T | Pre-recorded local based content about the city of Turin |
| Web Pages | HTML, PNG, GIF... | BTFTP over IPv6 over DVB MPE, DVB-T | Latest News |

Table 1: DVB-T Streaming Content

7.3 Web Broadcast over a DVB-T cell

The Web Broadcast over DVB-T is a service allowing to send Web pages (HTML, images, multimedia files in general) over a DVB-T cell, with both general and location based content; the external links are reached by the mobile UMTS network.

During the Demo, updated Web Pages were downloaded to the car using the DVB-T channel and the BTFTP protocol (see Annex C) over multicast IPv6. The contents, as they were downloaded, were cached locally in the IVAN Web Server. Every page was cyclically transmitted (Carousel transmission), giving the opportunity to update the pages.

7.3.1 Realization

The Web Pages are generated automatically from the Rai Teletvivo News. A server listens to the Teletvivo News and inserts them in a Database; an application retrieves some of the stored records and creates a set of linked HTML pages. To enrich the service and show location based information, some web pages on the city of Torino were used.

The BTFTP server gets the HTML pages with the last minute news and cyclically sends them over DVB MPE using the BTFTP protocol. From then on, the news are broadcasted to the local DVB-T cell

The following scheme clarifies the broadcast of the HTML pages over the DVB-T cell.

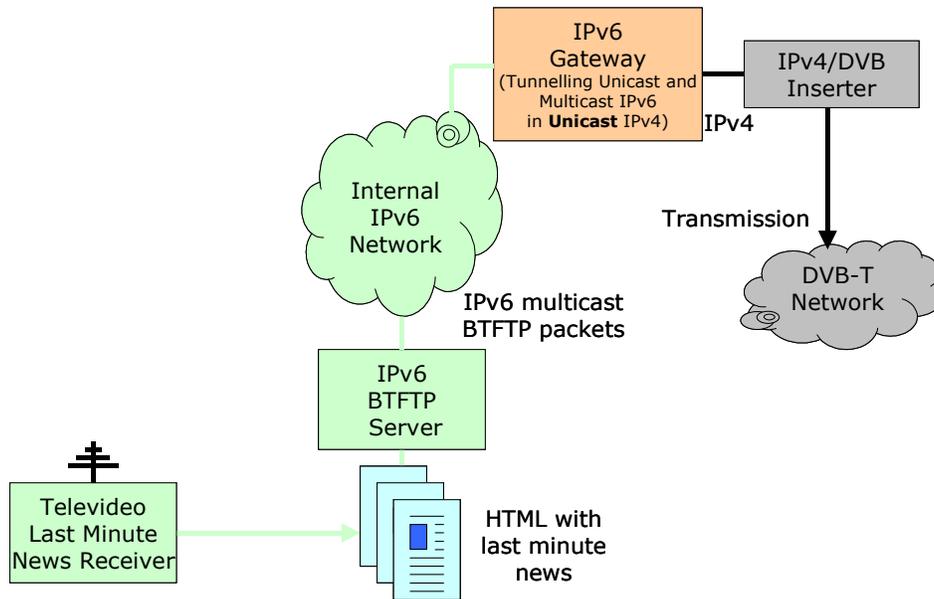


Figure 38: Basic architecture scheme used to broadcast Web pages over a DVB-T cell

7.3.2 IVAN Set Up

The car has DVB-T reception, so that the DVB-T Front-Box receives the HTML pages using a BTFTP decoder.

The following simple scheme clarifies the setup in the car.

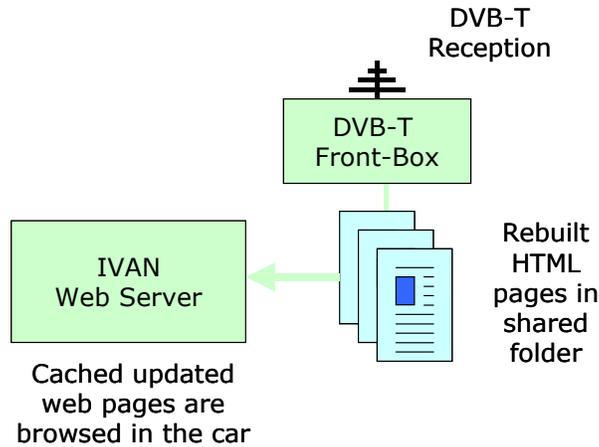


Figure 39: Reception of broadcast Web Pages over DVB-T

The Front-Box (see 8.2.1) stores the Web Pages in a local cache, and it shares the folder with the IVAN Web Server. So, users in the car can access the updated Web Pages.

7.3.3 Content Server

As described in the previous sections, Web Broadcast service makes use of a Server to retrieve information from the Rai Televideo News and to create the web page. Also a BTFTP Server is needed to send the pages on the DVB-T channel.

The contents for the Demonstration were based on the latest news, as they were sent on the Italian Televideo System. This way, the user in the car could access a set of latest, generic news without the need of a point-to-point connection. All the pages were sent using BTFTP over multicast IPv6. More detailed web pages could be reached following the web links; the UMTS connection allowed to navigate this extra information.

The software module allowing the reception of Web Pages included in the DVB-T Multiplex was the Java BTFTP Receiver.



Figure 40: Example Web Pages

The pictures in Figure 40 are examples of the Web Pages for the Demonstrator. The Web contents downloaded through the DVB-T channel are meant to contain generic and local based information (in this case last-minute news and information about the city of Turin). Every pages includes two kinds of links:

- local links: the target pages can be found in the local cache
- external links: the target pages are downloaded using the UMTS connection.

7.4 Multicast Messaging and Streaming

Two different types of services have been implemented to show and clarify the concepts of Multicast delivery services. The service can be selected via a small control application (Figure 41), which performs the necessary actions to send the session announcements to all clients and launches the actual sender applications.



Figure 41: Demonstration Control Application

The first service is a streaming service using multicast and unicast transmission technology. This service is selected via the “Multicast Streaming” and “Unicast Streaming” buttons (see Figure 41).

The streaming services was mainly used to explain the resource savings by transmitting only one stream Packet losses occur during the unicast streaming transmission and none during the multicast streaming transmission.

The streaming service was also used during the Group Partitioning demonstrations to have a more continuous transmission.

7.4.1 Realization

In case of Multicast Streaming, the stream is transmitted on an IP Multicast group to all clients simultaneously. In case of unicast streaming a predefined number of streams are sent from the streaming service (vlc) to the clients. The clients need to be registered (Group Membership function) to receive a unicast stream.

The Messaging demonstration services are implemented using RAI’s btftp protocol. The clients receive a stream of packets and store them locally in the file system. After the entire file was received, a notification is sent to the GUI signifying that new content is available. In case of Multicast messaging, the content flow is sent on an IP multicast group to all receivers simultaneously. A SAP announcement message is sent a few seconds earlier on the SAP control channel and triggers the reception process of the clients. In case of unicast messaging, each client gets its own copy of the file in via unicast.

7.4.2 Content Server

For the multicast streaming services, the VLC software from VideoLan.org has been used as server. VideoLan is able to stream mpeg2 content either via unicast or via multicast sessions.

For the multicast messaging service, the btftp software fro RAI as acts as content server. Based on the trigger from the Demo-Control Application (described in the overview section), the btftp software starts transmitting the specified file either via the unicast or via multicast sessions.

7.4.3 Content

| Content Type | Format | Transport | Short Description |
|-----------------------|--------|---------------------------------------|----------------------|
| Alice Audio/Vide | MPEG-2 | IP Unicast/ IP Multicast | 8min multimedia file |
| RAI Sports Collection | MPEG2 | Btftp using IP Unicast / IP Multicast | Short video clip |

7.5 Adaptive Video on Demand

The adaptive video on demand application demonstrated the use of a unicast service over different, abrupt changing link characteristics. In the context of the OverDRiVE Project an application has to deal with a heterogeneous multiradio environment with potentially hidden radio links. The user is able to change the quality of a video without restarting the video transmission and therefore to produce different amounts of traffic, according to the available bandwidth. As a further enhancement the client application was able to make its own decision about which quality level was appropriate, adapting to the changing network characteristics.

7.5.1 Realization

As streaming server a standard Darwin Streaming Server 4.1.3 [<http://developer.apple.com/darwin/projects/streaming/>] was used, which was patched to support IPv6. On the client side mpeg4ip 0.9.9 [<http://mpeg4ip.sourceforge.net/>] with extended functionalities was applied. Different MPEG4 video stream qualities of a video were stored in MPEG4 files. As depicted in Figure 42, the enhanced client was able to select via SDP ([18],[19] and [20]) between different average bitrates of the video causing different traffic loads.

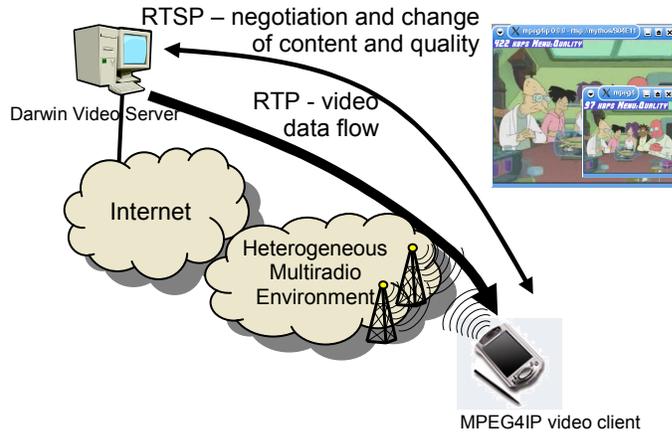


Figure 42: Adaptive Video on Demand

Automatic bottleneck link detection was handled via packet pair measurements [K. Lai, M. Baker, “Measuring Bandwidth”, in Proceedings of IEEE INFOCOM, Apr. 1999, pp. 235--245] and a loss based congestion control was applied. A detailed description of packet pair techniques can be found in D17.

7.5.2 IVAN Set Up

Streaming Server: Redhat 9.0 based Linux laptop in core network

Streaming Client: iPAQ 3870 running familiar Linux 0.7.1 and
laptop running debian Linux and mpeg4ip with mad audio codec and
adaptive streaming capabilities

7.5.3 Content Server

The Darwin Streaming Server 4.1.3 patched to support IPv6 was placed in the core test network.

7.5.4 Content

As content among other videos the Buena Vista music video clip “Chan Chan” was selected. A video clip is a valuable content for both video with sound and sound only – which should be shown while moving around and into the car and while using a low bandwidth GPRS/UMTS link. With broadband access in the car the user can pay attention to watching the video and enough bandwidth is available.



Figure 43: Chan Chan videos - 3 different qualities

7.6 Remote Access

Remote access of the IVAN provides basically the ability to query a web server located inside the vehicle and additionally to change the state of the web server. The web server allows web browsers to query some information about the IVAN, e.g. its connected devices and selected vehicle states, for example the position of its windows (opened or closed). Some of the states, for example the position of the windows are allowed to be changed from a browser, which allows an exemplified remote control of the vehicle.

7.6.1 Realization

Remote access is realized with an Apache IPv6 web server which runs inside the car. Besides showing an HTML page about vehicle information like speed, position of windows, and so forth, an HTML form will provide buttons to control the position of the car's windows. The web server can be accessed with a Laptop connected via GPRS, UMTS, Bluetooth, WLAN, or even with a smart mobile phone, which allows to run a web server.

Information that is shown by the web server:

- Current speed (mandatory)
- Mileage (mandatory)
- Position of windows (mandatory)
- Door lock state (optional)
- Revolutions per minute (optional)
- Temperature of water (optional)
- Ventilation, air conditioning (optional)
- Headlight, interior light (optional)

The web server allows to control the following functionality:

- Position of windows (mandatory)
- Door lock (optional)
- Ventilation, air conditioning (optional)
- Headlight, interior light (optional)
- Horn (optional)
- Flasher (optional)

NOTE: “optional” means, that the availability of this feature depends on the currently installed software revision, see section 7.7.

7.6.2 IVAN Set Up

Computer: CAN & web server (12V PC with 2 PCMCIA slots)

Operating System: Windows XP

Further Hardware: Network card, CAN interface card (both PCMCIA)

Further Software: CAN server, Apache web server

7.6.3 Content Server

Remote access is a service which is provided by the web server in the car, thus the content server in this scenario is equivalent to the server, described in the IVAN Set Up subsection.

7.6.4 Content

As mentioned before, the provided content consists in vehicle status information. The following two figures (Figure 44 and Figure 45) show an example screenshot, taken on a laptop that remotely accessed the car web server.

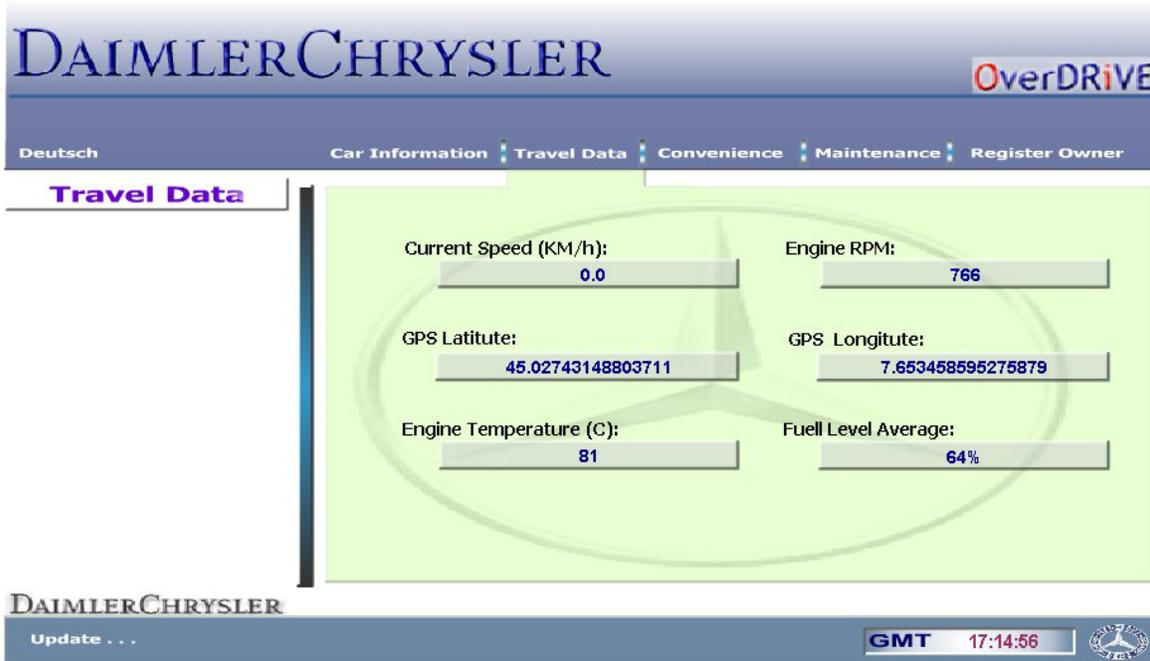


Figure 44: Remote Access - Travel Data

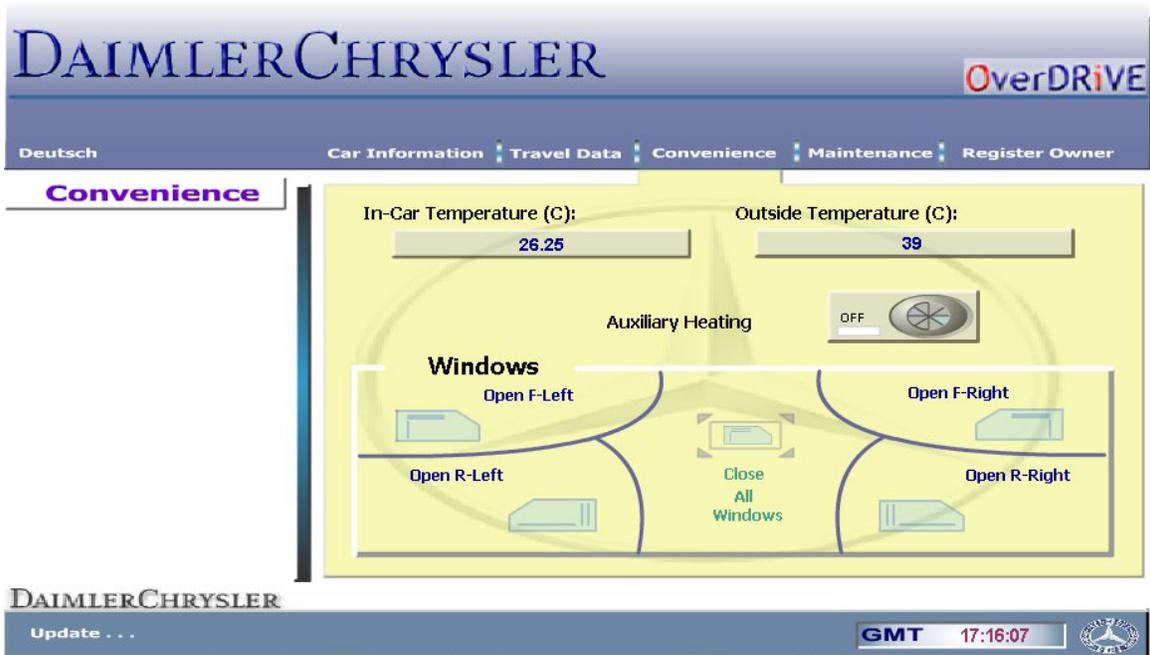


Figure 45: Remote Access - Convenience

7.7 Software Download

Software download shows the ability to increase the functionality of the in-vehicle IVAN or the vehicle’s functionality. We will demonstrate the principle by increasing the in-vehicle’s web server functionality by showing more information about the vehicle. After the software download, the user will be able to check the car lights and switch them on or off.

7.7.1 Realization

The software download is controlled by the web server, i.e. the web server provides a button to initiate a software download. After initiating the software download, the web server connects to an update server in the fixed infrastructure, to get the revised software component. Note, the revised software component is not really installed on the vehicle; rather it changes some states of the web server to reflect the correct software download and the correct operation of the windows after “installing” the software download (see Figure 46). This should be good for demonstrating the principle of a software download (proof of concept).

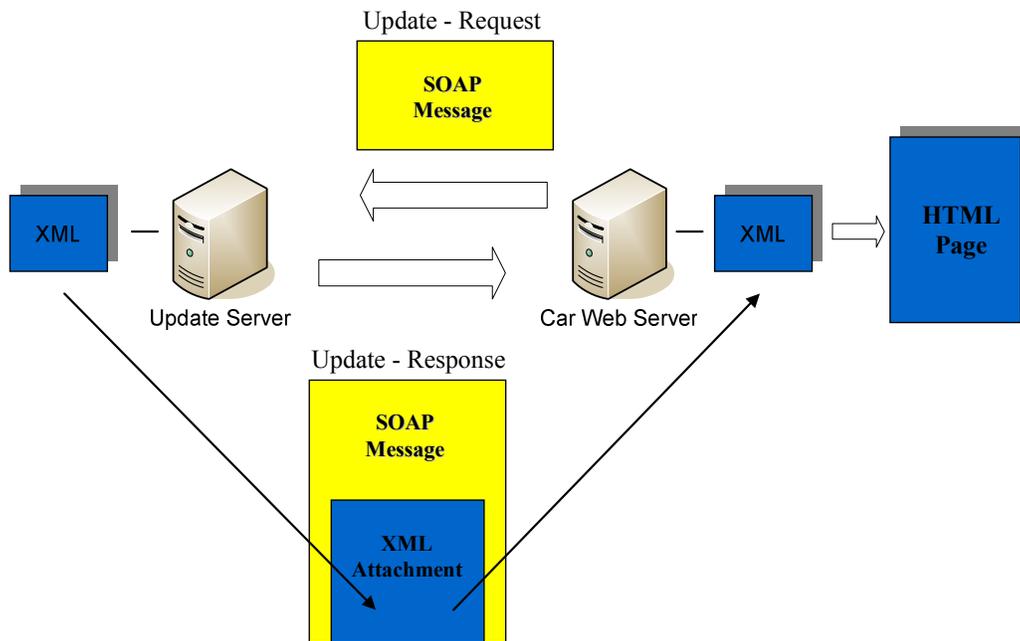


Figure 46: Remote Update with Java XML Messaging

The system is configured in a way that after rebooting the system, the old software configuration is in use, such that the software download can be done again (software downgrade, i.e. “installing” an old version, is also possible).

7.7.2 IVAN Set Up

Computer: CAN & web server in the car (as in the previous section), Laptop as update server

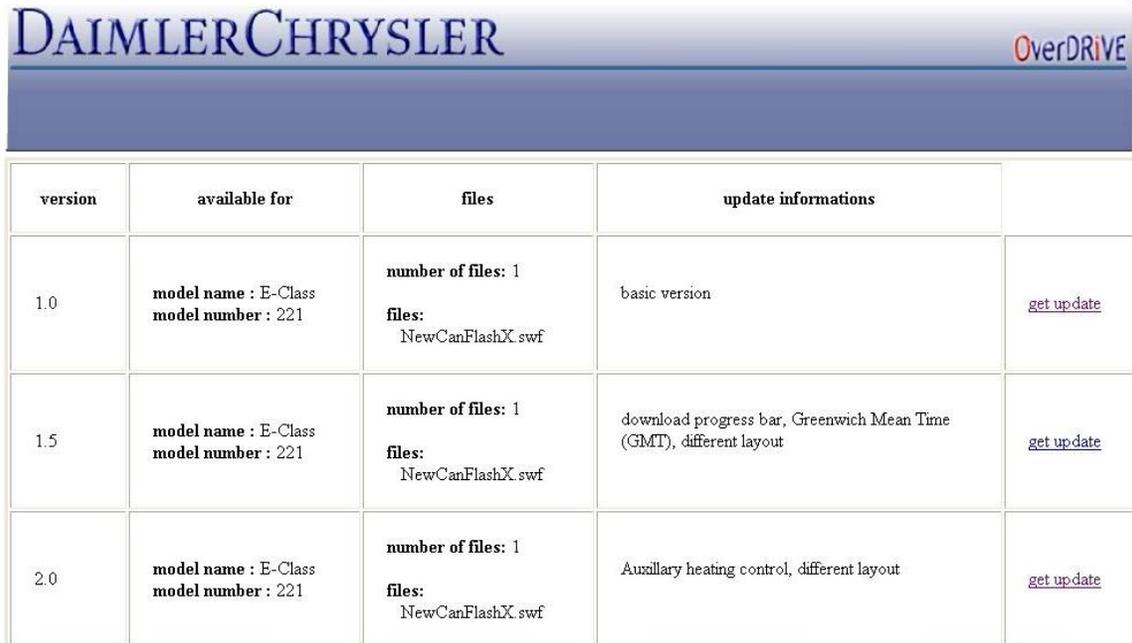
Operating Systems: Windows XP

7.7.3 Content Server

The Update Server that provides the software updates is located in the fixed infrastructure. It is running Windows XP and an Apache web server.

7.7.4 Content

The Update Server offers different versions of the car web server interface template. Figure 47 shows an example. Apart from the update itself, the server shows also additional information on the update, i.e. the provided improvements.



| version | available for | files | update informations | |
|---------|--|--|--|----------------------------|
| 1.0 | model name : E-Class model number : 221 | number of files: 1 files: NewCanFlashX.swf | basic version | get update |
| 1.5 | model name : E-Class model number : 221 | number of files: 1 files: NewCanFlashX.swf | download progress bar, Greenwich Mean Time (GMT), different layout | get update |
| 2.0 | model name : E-Class model number : 221 | number of files: 1 files: NewCanFlashX.swf | Auxillary heating control, different layout | get update |

Figure 47: Remote Update

7.8 Web Access

Browsing the World Wide Web is one of the most commonly used services in the Internet. This service is provided by web servers via the HTTP protocol. Unfortunately not all web servers offer support for IPv6 and not all domains are connected to the 6bone [www.6bone.net/]. This means that some content is only accessible by using the IPv4 protocol. On the other hand the homepage of the KAME project [http://www.kame.net/] offers a moving image “dancing kame” only visible to people accessing the page via IPv6 to visualize the use of IPv6. The web browsing service demonstrated was able to do both: on the one hand access to IPv6 enabled web servers via IPv6 and on the other hand access to IPv4 only web servers via IPv4.

7.8.1 Realization

The IVAN of the demonstrator car is an IPv6 only network. This means that only IPv6 traffic is routed by the mobile router. As a consequence clients within the IVAN can only access IPv6 enabled web servers connected to the 6bone. It is believed that in addition to new services the audience of the demonstration may expect to use clients in the IVAN the same way they commonly do. This means in the context of web browsing that they want to visit their preferred web sites which may be IPv4 only.

For this reason a light-weight HTTP proxy had to be used. This proxy would have to map the IPv6 requests from clients to IPv4 as shown in Figure 48. The proxy does not need to support the

caching of web sites. This proxy could be located anywhere in the Internet where it has both IPv4 and IPv6 connectivity.

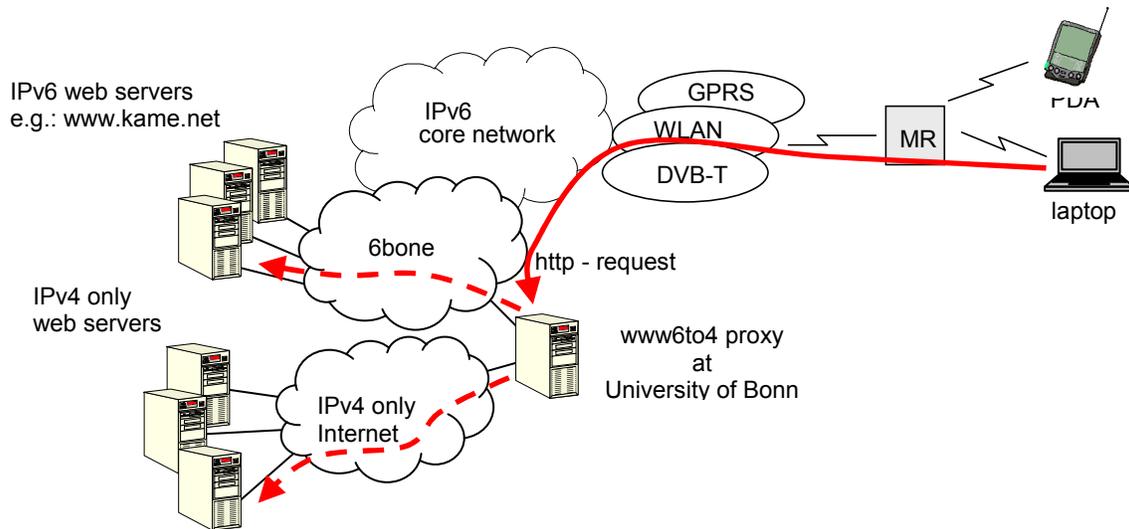


Figure 48: HTTP-proxy demo setup

Candidates were Squid (www.squid-cache.org/), todt, www6to4 and wwwoffle (all available on <http://www.vermicelli.pasta.cs.uit.no/IPv6/software.html>). Because of its simplicity and easy configurability www6to4 was chosen. www6to4 was originally designed as a front-end to enable IPv4 only browsers to access IPv6 content but also works as a stand-alone proxy for IPv6 clients. Clients connect to the www6to4 proxy via IPv6 or IPv4. The www6to4 proxy parses the request and establishes a connection to the web server based on the DNS entry of the server either via IPv4 or IPv6.

On the client side an IPv6 enabled web browser is needed that also supports IPv6 proxies. Mozilla (<http://www.mozilla.org/>) directly accepts IPv6 addresses as input. The Microsoft Internet Browser does not parse IPv6 addresses of proxies correctly. However, a host name can be entered in the hosts file that correspond to an IPv6 address. As www6to4 does not support the persistent connections of HTTP/1.1 [RFC 2068] the browser has to be configured accordingly.

A small patch was developed to be able to install the www6to4 proxy on a Linux machine. The reason was that the standard Linux stack is a dual stack architecture [21]. www6to4 tried to open the same port once for IPv4 and then again for IPv6.

The simplicity of configuration made it possible for the partners to install the proxy in their local test bed. For the demonstration www6to4 was accessible at the University of Bonn on a Linux machine.

7.8.2 IVAN Set Up

Web proxy: Suse 8.2 based Linux PC at University of Bonn, Germany

Web clients: Laptop running debian Linux and Mozilla browser
iPAQ 3870 running familiar linux 0.7.1 and dillo browser

7.8.3 Content Server

As this service is meant to provide general web access, no special content servers were installed. Visitors of the demonstration were able to access any IPv6 and IPv4 web servers.

7.8.4 Content

Again, as this service is meant to provide general web access, no special content was installed. However the Project homepage of the OverDRiVE project www.ist-OverDRiVE.org/ might be of special interest. Another interesting homepage is the already mentioned homepage of the KAME project [<http://www.kame.net/>] which visualizes IPv6 access.

8 External Interfaces

8.1 Network Architecture

8.1.1 6Bone, architecture figure and description

Real time connection to the WWW and to other IPv6 sites requires a link with the external 6Bone. The 6Bone allows to reach IPv6 enabled internet sites and services.

Rai configured the access to the external network using IPv6 over IPv4 tunnelling and a Tunnel Broker. The following picture clarifies the architecture used at Rai Crit for the connection to the 6Bone. Some sample servers have been put in the picture; however it's not the complete architecture scheme, of course.

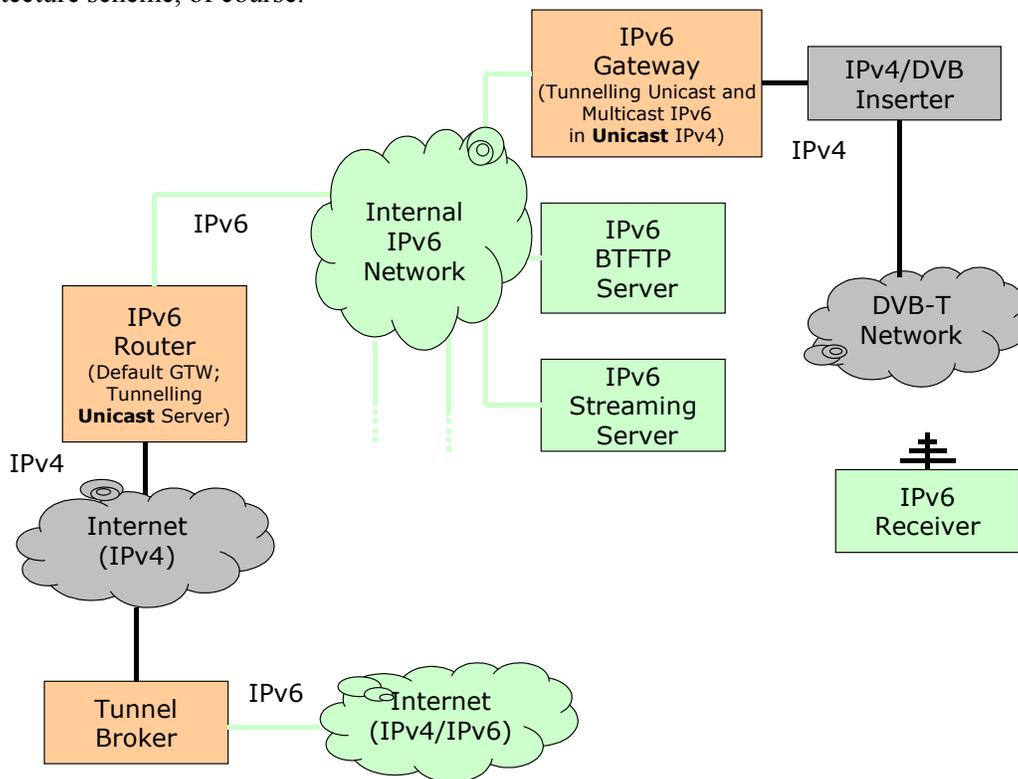


Figure 49: Access to the 6Bone from the Rai Crit site, during the demo

The IPv6 Router is a router allowing IPv6 over IPv4 tunnelling to the Tunnel Broker. The functions of the Router are two:

- connectivity to the 6Bone through an IPv6 over IPv4 tunnel
- distributing IPv6 public addresses to all the local network, using Router Advertisements.

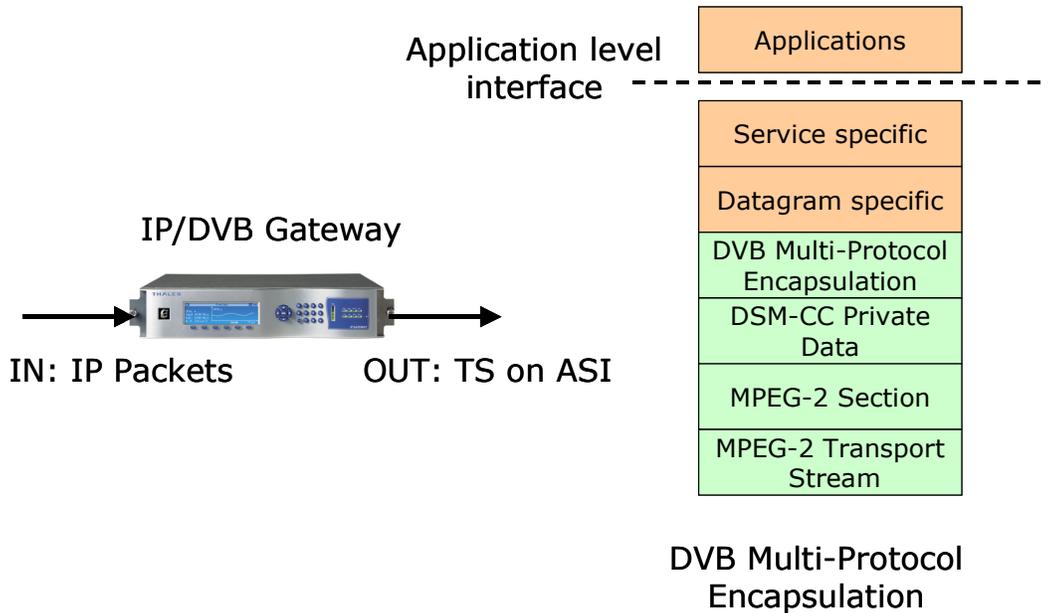
The total throughput of the network is limited by the available bandwidth of the Tunnel Broker and the available internet connection.

The IPv6 Gateway provides routing and tunnelling functionalities: it takes the IPv6 multicast packets from the LAN, it encapsulates them in IPv4 packets and sends these to the IP/DVB Gateway, for DVB-T transmission.

The IPv6 enabled receiver is in this case the Front-Box inside the car.

8.1.2 IP/DVB Gateway

The equipment allowing the transport of IP packets in a DVB-T Transport Stream is the IP/DVB Gateway. The following picture shows the Gateway (on the left) and the protocol stack that complies with the DVB MPE specification [13, 14] (on the right).



The operator configures the IP/DVB Gateway

- adding some filters corresponding to the source/destination IP addresses of the packets that must be encapsulated
- setting the DVB parameters of the transport stream to be generated (PIDs, bitrate)
- customizing the DVB tables (PMT, SDT...) included in the TS

When a packet with the right IP address reaches the Ethernet interface of it and satisfies the requirements that the operator specified, the Gateway inserts the packet in the DVB stream, according to the DVB MPE Profile.

At the time the demonstration was made, there were no IPv6 enabled IP/DVB Gateways available on the market, so it was necessary to insert the IPv6 packets in IPv4 packets, using a SIT Tunnel [15].

8.2 DVB

8.2.1 DVB Front box

The broadcast DVB-T channel is well suited for the transport of general interest information to a large number of users. The DVB MPE specification allows the encapsulation of IP datagrams in

the DVB transport stream. Nowadays a lot of IPv4 DVB-IP Gateways exist and they allow a seamless transport of IP streams on DVB channels.

A different situation occurs for IPv6. While IPv6 is actually supported by the specification, it's not yet implemented by DVB-IP Gateways available on the market. So, IPv6 packets have to be encapsulated in IPv4 packets before they are sent to the DVB-IP Gateway and transmitted on the DVB-T channel.

Ever since OverDRiVE will use the broadcast channel to transfer IPv6 data with multicast addressing, we're considering in the following sections the transmission of multicast IPv6 packets, with the help of a unicast IPv4 tunnel.

The DVB Front-Box is an equipment that allows the decapsulation of IPv6 packets from the IPv4 tunnel, forwarding them in the local LAN (see next figure).

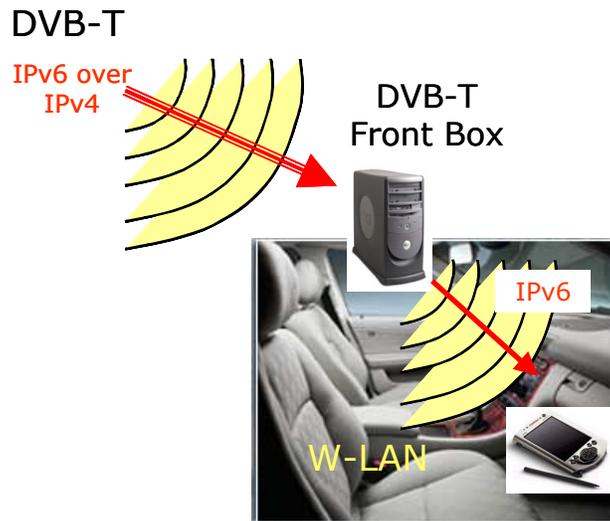


Figure 50: DVB-T Front Box, receiver side

The DVB-T Front Box is a PC with an operating system based on Linux Redhat 9.0. The main hardware components are:

- an Ethernet NIC for the forwarding of the received packets
- a TechnoTrend DVB-T Reception Card (with an integrated MPEG-2 decoder, version 1.6)

The installation of the DVB-T Card implies a recompilation of the Linux Kernel (the 2.4.20 release of the kernel was used in this case) and the installation (and ad hoc patching – the Card ID of this particular model is not automatically recognized by the drivers) of the Linux DVB Drivers drivers (<http://www.linuxtv.org>).

The DVB-T MPEG-2 audio/video is decoded by the integrated decoder of the DVB-T Card, and an helper application selects the right frequency and parameters for the tuning. The check of the audio/video content is made by a rendering application (e.g. xawtv).

The MPE encapsulated IPv6 in IPv4 packets are easily extracted to a virtual network interface, `dvb0_0`. A SIT tunnel makes the extraction of the multicast IPv6 and the problem at this point is that Linux Kernel 2.4.20 doesn't support static multicast IPv6 routing at the moment, so the packets remain in the DVB-T Front Box.

8.2.2 Forwarding the received IPv6 packets: fwdsix

The IPv6 packets have to be transmitted on the local LAN from the DVB-T Front Box, so OverDRiVE has developed for this purpose an application, *fwdsix*, providing a temporary solution for the tests. This application allows to extract the multicast IPv6 from the SIT and forward them to the LAN of the DVB-T Front Box, without the need of the support of a routing protocol. The advantage of this solution is that is very simple to extend and debug the application, and the forwarding decision is made in the DVB-T Front Box itself, that becomes the “service provider” of its local LAN. The forwarding with this custom application is not standard because it doesn’t implement any real routing protocol, behaving like a repeater.

The final solution should involve the use of a standard routing protocol (of course, the implementation of the protocol was not available at the time the Demonstration was created).

8.2.3 Architecture description

The architecture used for the transmission of the IPv6 packets on the DVB-T network is based on the following components:

- The dual NIC IPv6 over IPv4 Encapsulator
- An IP/DVB Gateway (that won’t be described here)
- The DVB-T Transmission chain (that won’t be described here)
- The DVB-T Front Box, that receives and the DVB-T signal

The IPv6 multicast contents are generated on the Service Provider’s LAN. An IPv6-over-IPv4 Gateway encapsulates the IPv6 Packets in IPv4 unicast packets.



Figure 51: DVB-T Front Box, receiver side

At the receiving side, the DVB-T Front Box receives the DVB-T signal, and extracts the DVB MPE encapsulated unicast IPv4 packets. The DVB-T Front Box is the end point of the IPv6 over IPv4 tunnel. The multicast IPv6 packets are then forwarded to the mobile LAN in the car.

An important note is that *there’s no return channel*: the multicast packets are sent in a uni-directional way. This allows the use of simple audio and video streaming (e.g. with VideoLAN) and the transmission of data (with the BTFTP protocol).

8.2.4 IPv6 over IPv4 Encapsulator Setup: mproxy

The IPv6 over IPv4 Encapsulator is the component that allows to select relevant multicast IPv6 packets from the transmitting LAN, and to encapsulate them in IPv4, acting like an adapter between the IPv6 contents and the IP/DVB Gateway. Theoretically, once IPv6 enabled DVB Gateways will be commercially available, this component will be unnecessary.

The IPv6 over IPv4 Encapsulator is a Dual NIC PC with a Linux Redhat 9.0 Operating System. The encapsulation is made through a SIT IPv6 over IPv4 tunnel, without the need to install additional software.

The VideoLAN vlc server, generates the multicast IPv6 packets to be encapsulated on IPv4. The multicast packets are streamed on the LAN and sent to the Encapsulator

An application called mproxy has been developed by OverDRiVE, listening to multicast IPv6 packets and encapsulating them in a IPv4 SIT tunnel. For this purpose a unicast IPv4 address was chosen: SIT tunnels don't support at the moment the use of multicast IPv4 addresses.

8.2.5 DVB Platform

In February 1998 the RAI Research and Technology Innovation Centre (CRIT) in Turin started up the first experimental DVB-T transmissions in Italy. The digital platform was further enriched to include a Data Broadcasting Centre providing MHP multimedia applications and data broadcasting services.

The source encoding and multiplexing chain consists of six statistical multiplexing encoders, one re-multiplexer to provide EPG and MHP applications, and an SI-PSI generator. The multiplexer receives the Packetised Elementary Stream (PES) from the encoders and the multimedia data from the Data Broadcasting Centre, and generates two Transport Streams at the bit-rates of 6.03 Mbps (or 12 Mbps when a more efficient modulation is adopted) and 24.13 Mbps, each one delivering a DVB-T bouquet of services and applications. A PC controls the parameters of the encoding and multiplexing chain as shown in Figure 52.

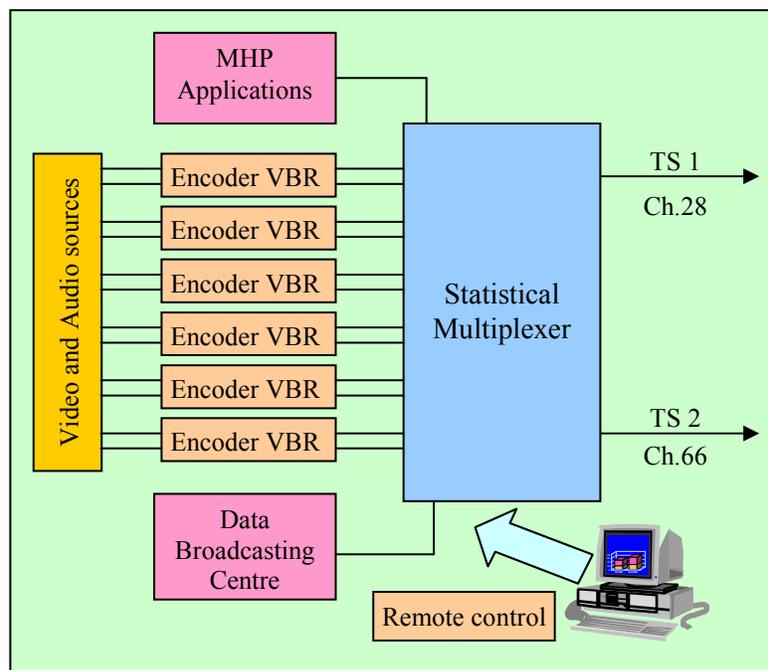


Figure 52: DVB encoding and multiplexing chain

The Transport Streams are transferred to the main transmitter – situated on a hill 500 m a.g.l., 5 km away from the CRIT – by means of a 45 Mbps tributary of an SDH digital radio link. Two ATM Network Adapters are used to convert the multiplexer outputs to the transmission bit-rate.

At the transmitter the two digital streams are re-converted to the original bit-rates of 6.03 Mbps (or 12.06 Mbps) and 24.13 Mbps and sent to the COFDM modulators of two independent UHF transmission chains, one operating on channel 28 (530 MHz) and the other one on channel 66 (834 MHz). Table 2 gives the modulation and r.f. parameters for both chains.

| Ch. | Modulation | Bit-rate (Mbps) | TX (W) | ERP (W) | Pol |
|-----|-------------------------------|-----------------|--------|---------|-----|
| 28 | QPSK - 16QAM 2k; 1/2; 1/32 | 6.03 - 12.06 | 40 | 300 | V |
| 66 | 64QAM 8k; 2/3; 1/32 | 24.13 | 350 | 2000 | H |

Table 2: Modulation and r.f. Parameters

The modulation scheme used on channel 28 is very robust and is therefore an appropriate choice for **mobile reception** trials and for reliable indoor portable reception. The use of vertical polarisation is also appropriate for mobile services. The bouquet of programmes is configured as in Table 3. For the Demonstration the contents have been customized.

It includes one TV programme (or three programmes, including a regional programme if a more efficient modulation scheme is adopted) and about 1.8 Mbps of Data.

| Bouquet | Bit-rate | |
|--------------------|----------------|----------|
| | Video | Audio |
| Rai Uno | 4 Mbps*, VBR | 192 kbps |
| Rai News 24** | 2.5 Mbps*, VBR | 192 kbps |
| Rai Tre Piemonte** | 3.5 Mbps*, VBR | 192 kbps |
| DATA | 1.8 Mbps | |

Table 3: Channel 28 Bouquet

* Medium value

** Present in the bouquet only if a more efficient modulation is adopted

The modulation scheme used on channel 66 is very efficient in terms of capacity and allows transmitting up to 5 TV programmes (see Table 4). This configuration, which provides a wide and diversified offer of services, is then particularly oriented to fixed and portable reception in urban areas.

| Bouquet | Bit-rate | |
|------------------|----------------|----------|
| | Video | Audio |
| Rai Uno | 5 Mbps*, VBR | 192 kbps |
| Rai Due | 4 Mbps*, VBR | 192 kbps |
| Rai Tre Piemonte | 4 Mbps*, VBR | 192 kbps |
| Rai Sport | 5.5 Mbps*, VBR | 192 kbps |
| Rai News 24 | 2.5 Mbps*, VBR | 192 kbps |

21 Mbps
statistical
multiplex

| | |
|------------------|--------|
| MHP Applications | 2 Mbps |
|------------------|--------|

Table 4: Channel 66 Bouquet

* Medium value

The two UHF transmitters on channels 28 and 66 make use of different antenna systems which are installed on the same mast.

Since the radio frequency spectrum adjacent to channels 28 and 66 is occupied by analogue TV services, both DVB-T transmitting chains include suitable r.f. output filters to guarantee adequate protection versus the analogue services (DVB-T vs. PAL). The same r.f. filters ensure adequate rejection on the taboo channels, relevant to both channels 28 and 66, which are currently occupied by analogue services in the same area.

During the final demonstration of OverDRiVE the following bouquet has been broadcasted (see Table 5):

| Bouquet | Bit-rate | |
|---------|---------------|----------|
| | Video | Audio |
| Rai Uno | 2,5 Mbps, CBR | 192 kbps |
| Torino | 1.5 Mbps, CBR | 192 kbps |
| IP DATA | 1.5 Mbps | |

Table 5: Bouquet for the Final Demonstration of OverDRiVE

The bouquet reported in Table 3 was adopted for both the channels 28 and 66 but during the demonstration only the channel 66 was used because it interfered less than the other one. Moreover the ERP of channel 66, as shown in Table 1, is much greater than the one of channel 28 and so a very good coverage was guaranteed.

The modulation scheme was the following (Table 6):

| Ch. | Modulation | Bit-rate (Mbps) |
|-----------|-------------------------|-----------------|
| 28 and 66 | QPSK - 2k; 1/2; 1/32 | 6.03 |

Table 6: Modulation Scheme for the Final Demonstration of OverDRiVE

The test carried out before the demonstration has proved the reliability of mobile reception on channel 66 at least with the robust modulation above indicated.

8.3 GPRS and UMTS

8.3.1 GPRS

This section presents the description of mobile multicast support for GPRS terminals in the OverDRiVE testbed. The main goal of supporting GPRS technology at the Mobile Router (MR) is to provide an uplink connection for DVB-T downlink sessions, and demonstrate the intersystem handover between WLAN (the “default” access in the testbed) and GPRS technologies.

There were several reasons to include GPRS in the OverDRiVE testbed. At the beginning, within OverDRiVE we did not plan to have an operable public UMTS network. We had access only to WLAN (802.11b), GPRS, and Bluetooth, as candidate technologies for the demonstration testbed.

Among these, Bluetooth is not suitable for long-range communication. WLAN was the basic technology in all testbeds developed within OverDRiVE, and was explicitly referred in the project goals. On the other hand, GPRS is considered a 2.5G mobile technology, a bridge between GSM and UMTS.

Besides the above described technologies, we used DVB-T based download links. However, DVB-T is a special broadcast technology, that does not provide an uplink connection. Therefore, we needed a technology that provides this connection, and is available even outside WLAN hotspots.

Based on these considerations, we have chosen public GPRS as a technology to be supported by the testbed.

The GPRS interface was not the only one enabled at the mobile router. Either it was used together with the DVB-T interface, or the WLAN one, not to mention the interfaces selected to support access routers and/or end terminals within the MONET. Also, in order to ensure parallel development, a modular development is useful. In that way, the development of the GPRS support module did not affect the development of the MR and vice-versa. According to the selected OverDRiVE terminology, we refer to the GPRS module as the *GPRS FrontBox*.

- Why is it good? – It separates GPRS and IPv6-to-IPv4 tunnelling at the MR
- How does it work? – It uses a GPRS modem built in a mobile phone, and uses IPv4/IPv6 tunnelling.
- Availability? – Every Linux machine with Serial Port and ETH interface can implement it.

GPRS FrontBox Architecture

The GPRS FrontBox architecture can be split in two parts. First, the GPRS interface should be enabled using a GPRS-capable mobile phone and the GPRS subscription. Figure 44 presents the schematic setup of the GPRS FrontBox.

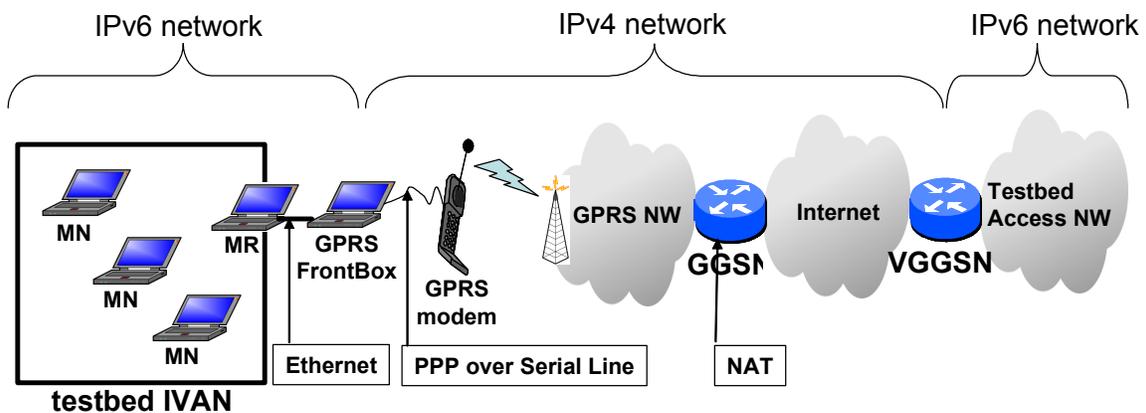


Figure 53: The GPRS FrontBox Architecture

Then, since the operators do not support IPv6 in their networks and the IP addresses assigned to GPRS subscribers are NAT-ed at their GGSNs, we have to introduce an additional network element, called *Virtual GGSN*, or VGGSN. The VGGSN is placed in the testbed, and has a native IPv6 interface towards the testbed routers. It acts as the GPRS gateway of the testbed, i.e., it routes all traffic to and from the GPRS interfaces of the mobile terminals through its second IPv4 enabled interface, connected to the public internet.

The mobile phone

The GPRS FrontBox uses a GPRS enabled GSM mobile terminal. The terminal in our case is a phone. Nevertheless, with minor changes, a PCMCIA card can be supported as well. Current GPRS enabled PCMCIA cards cannot be configured as interfaces (at the likes of a PCMCIA Ethernet interface). The PCMCIA-based GPRS devices have to be reached through a PPP connection, similar to the GPRS phones. Therefore, the only difference between the phone and the PCMCIA card based solution is the establishment of the physical connection between these devices and the FrontBox.

In our case the PC must establish a serial line connection to the mobile phone. For this purpose we use a serial interface cable. If the FrontBox has enabled its serial interface, it is enough to use the standard kernel support included in every current Linux distribution. The connections between the different elements of the GPRS supporting architecture are presented in Figure 45. Connection 1 represents this serial connection.

We manage the GPRS modem of the mobile phone through the Point-to-Point Protocol (PPP). Furthermore, we use the FrontBox-side of the PPP interface to reach the GPRS network. The PPP module in Linux is implemented via the *pppd* user space daemon. The configuration is controlled by the */etc/ppp/options* file. The *pppd* will establish a connection between the FrontBox and the phone (connection 2 in Figure 45), bring up a */dev/pppX* interface in the FrontBox (X=0 in our case) and it will configure IPv4 addresses at both extremities of the link. The *pppd* daemon will run a short script with the codes supplied by the GPRS operator (*/etc/ppp/gprs_chat*) to initialize the GPRS connection between the phone and the GGSN (connection 3 in Figure 45).

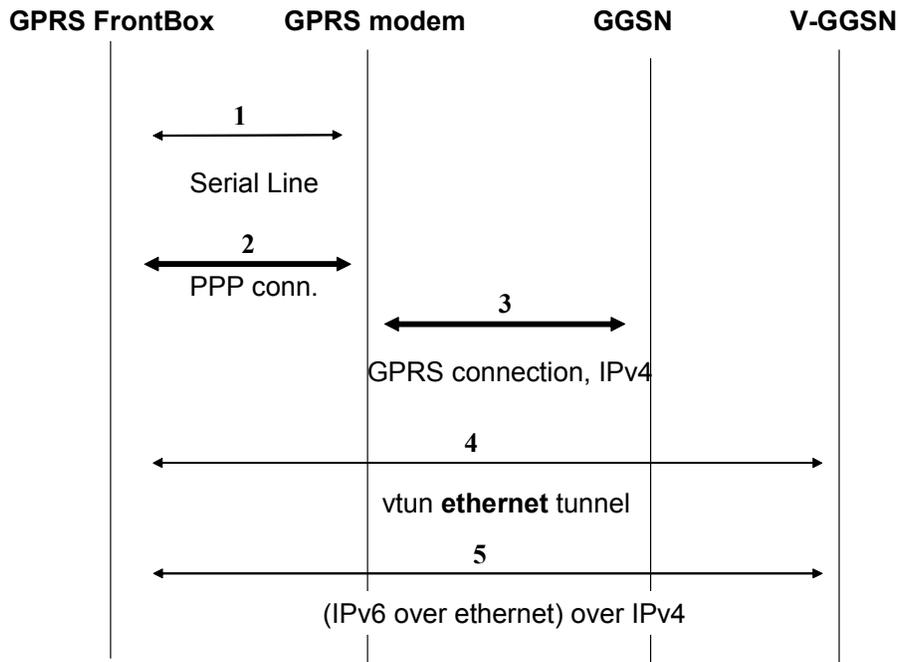


Figure 54: The connections in the architecture

By now, we have an established connection between the PC and the GPRS phone; the */dev/pppX* interface is up and the GPRS service is initialized.

The FrontBox

As we mentioned in the description of the architecture, we set up a special networking element in the testbed network, the Virtual GGSN (VGGSN). The VGGSN is the gateway at the Access Network for every packet received from the MR via the GPRS interface. This also means that the VGGSN should handle the MIPv6 signalling. However, we have found that the tunneling software is unstable with several MIP versions. To avoid this problem, we use the VGGSN as a proxy. Therefore, the VGGSN will be a ‘FrontBox’ at the Access Network side, and its second Ethernet interface will work as the “delegate” of the MR’s GPRS interface.

To forward packets across the GPRS IPv4 segment (see Figure 44), we use a tunnel between the FrontBox and the VGGSN. Since we do not know in advance the IP address assigned by the GPRS operator, and due to the NAT done at the GGSN, we have to use a tunneling approach based on a master-slave model. Linux offers the *vtun* module for such needs.

The kernel version 2.4.21 has been used with the Debian distribution, with IPv6 options enabled. The installation of the *vtun* requires the *lzo*, *zlib*, and *openssl* packages, besides the above mentioned ones. On both the GPRS FrontBox and the VGGSN a special device IO file was created. The */dev/net/tun* is a special file, with major 10, minor 200, and it enables char type operations. None of the two PCs should run MIP codes, since the current *vtun* modules hardly interoperate with them.

The */usr/local/sbin/vtund* starts the tunnelling module and the configuration commands are stored in the */usr/local/etc/vtund.conf* file. The resulted tunnel (connection 4, Figure 2) is an *Ethernet* tunnel, where the encapsulated IP packets are those sent by the CN and the MR. This results in Ethernet over IPv4 packets. These packets should be further encapsulated into IPv6 packets (connection 5, Figure 45), by usual IPv6 over ethernet encapsulation. The *vtun* should be set up as a server at the VGGSN. The command *vtund -s -p 5000* starts the *vtund* in server mode at port 5000. At the FrontBox the *vtund* will be run in client mode. To avoid typing all the parameters, the file */etc/vtund-start.conf* may contain the necessary data: the IPv4 address of the VGGSN and the *vtun* session name. Then, the */etc/init.d/vtund start* command starts the tunnel setup. The tunnel will result in a virtual Ethernet interface at both machines, named *tap0*. According to our experience, from the moment of issuing the *pppd* command till the *tap0* interface is initiated takes around 10 minutes. Finally, the routing should be set up to route all packets for the mobile terminals through the GPRS tunnel at the VGGSN. At the GPRS FrontBox we can use a routing based forwarding, if the subnet of the MR is straight-forward (the packets with the destinations from the subnet are sent to the MR, the rest to the tunnel). Otherwise, we may use an IPv6 forwarding engine that ‘proxies’ the traffic from the two interfaces: everything coming from interface A is sent to interface B, and vice-versa.

IPv6 forwarding (Relay)

To assure the forwarding of IPv6 packets such as MLD messages, multicast packets, and MIPv6 signalling, we developed a *Relay* program. A similar program has been developed by RAI for their DVB-T FrontBox as well..

Implementations for the PIM-SM protocol suit for IPv6 only exist for BSD systems (KAME project), while other networking options are easier to implement under Linux. Therefore, our access points virtually consist of a Linux-AP and a BSD multicast router. It implies that mobile hosts do not connect directly to the last multicast router, but via a Linux AR. As Multicast Listener Discovery messages are sent with TTL=1 using link local addresses, we needed a utility on Linux APs that forwards MLD messages between the multicast router and the mobile host.

Program description

The *Relay* program opens 2 raw sockets (see *man packet* for details), one bound to the uplink and the other bound to the downlink interface of the host (interface names are specified as command line parameters). Both sockets only listen to Ethernet packets with IPv6 payload. The *Relay* program reads and filters all these incoming packets.

MLD packets coming from the uplink interface are forwarded to the downlink interface and vice versa. Multicast UDP packets are only forwarded if coming from the uplink. In our test we needed only UDP multicast packets as data. By this mechanism we filtered out any possible status information that might trigger a mis-configuration of the FrontBox. All other packets are ignored. Packets are forwarded unchanged, except for Ethernet source addresses. These are replaced by the MAC address of the forwarding interface, because WaveLAN cards can't send the packet if its source address differs from the MAC address of the card.

In this way, bidirectional flow of MLD signalling and downlink multicast data forwarding is ensured between mobile hosts and the multicast router.

8.3.2 UMTS

This section describes the differences with respect to the UMTS dial-up procedure compared to GPRS dial-up. Basically UMTS behaves the same as GPRS when looking at IP address handling and Internet access. As in the GPRS case only IPv4 private addresses were provided. With the above described frontbox tunnelling solution native IPv6 services could be used in our demonstrator setup.

The setup of the dial-up requires in contrast to GPRS some more AT commands to be sent from the client to the mobile. The finally working setup including init-scripts can be found at [http://www.comnets.rwth-aachen.de/~o_drive/software.html].

8.4 WLAN

WLAN (802.11b) is the primary uplink system that is used during the demonstration. The WLAN connectivity between the IVAN and the infrastructure is realized by two WLAN access points (APs) in the infrastructure and WLAN PC-Card with an external antenna inside the car's mobile router. The different access networks (called home and foreign, see Figure 19: Demonstration Setup) are selected by changing the WLAN ESSID on the mobile router.

On the mobile router, the PC-Card is supported by standard Linux drivers, which means it can be configured by the common tools “iwconfig” and “ifconfig”. The first is used to modify WLAN specific parameters such as the ESSID or the encryption key. The latter is used to display and change interface related parameters such as IPv6 address or network mask.

The access points are configurable via web interface or telnet interface (depending on the manufacturer).

9 Internal Interfaces

9.1 IVAN Architecture

The IVAN architecture is shown in the following figure:

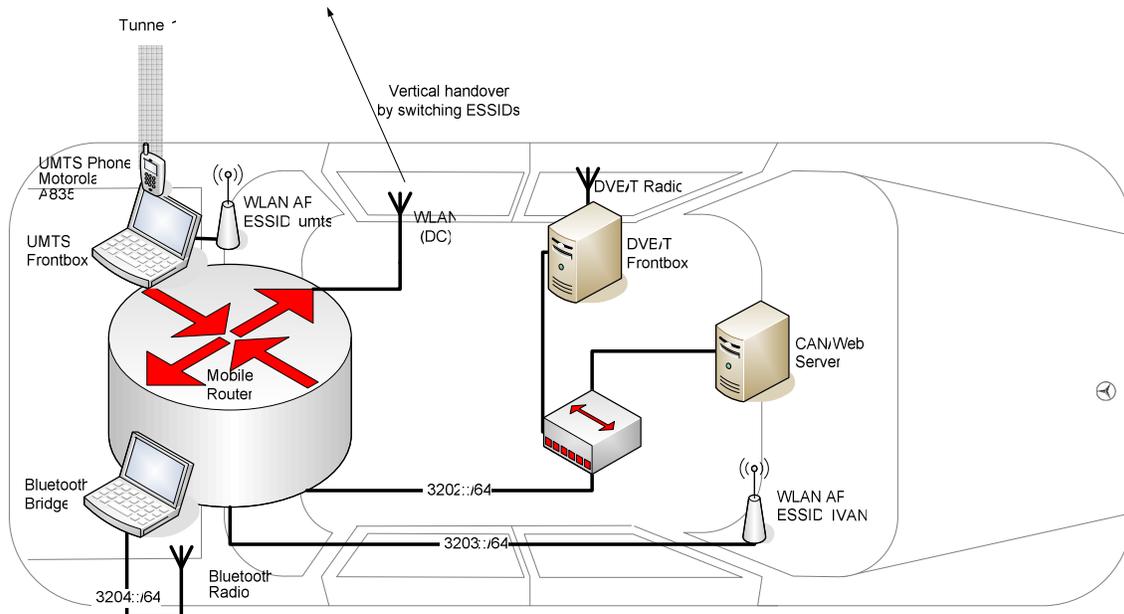


Figure 55: IVAN Architecture

Basically, it consists in of the following devices:

- Mobile Router
- CAN/Web Server
- Internal WLAN Access Point
- Bluetooth Bridge
- And UMTS Frontbox + UMTS Phone + WLAN Access Point
- DVB/T Frontbox

The following sections give a more detailed description of these components.

9.2 CAN Bus interface

The CAN (Controller Area network) is the central bus for controlling of most of the car's functionality with respect to engine management, interior devices, sensors and actuators (NOTE: multimedia and most of the telematics function are handled by the MOST bus system). Controller Area Network (CAN) is a serial communication protocol that may be used to transfer up to 8 data bytes within a single message. For larger amounts of data, multiple messages are commonly used. Most Controller Area Network (CAN)-based networks select a single bit rate. While communication bit rates may be as high as 1 M BPS, most implementations are 500K BPS or

less. Controller Area Network (CAN) supports data transfers between multiple peers. No master controller is needed to supervise the network conversation. The Controller Area Network (CAN) message is bit-oriented, always begins with a "start of message" indication, includes an address (called identifier), may contain data, includes a CRC, requires an acknowledgment from all network members, and is converted to an appropriate signal by the selected physical layer before being placed on the shared media (typically wires). Controller Area Network (CAN) provides a simple mechanism, called bitwise arbitration, to eliminate competing transmitters from colliding into each other during message initiation.

The access to the CAN bus is realized using a Softing CAN PCCard with the appropriate drivers for Windows operating system. DCAG provided a CORBA based interface for allowing other programs to read and write CAN values. That interface was used to realize the CAN interface application.

9.3 Bluetooth

In parallel to WLAN Bluetooth devices were used to connect mobile nodes (iPAQs and notebooks) to the IVAN. All involved components that used Bluetooth were running Linux with the BlueZ stack [<http://www.bluez.org/>].

As communication profile PAN was used [<http://bluez.sourceforge.net/contrib/HOWTO-PAN/>]. This allows for the transport of IEEE 802.3 frames over Bluetooth and opens the usage of a verity of protocols (IPv4, IPv6, IPX, ...).

In the demonstrator setup, the role of a PANU (PAN User) was dedicated to the mobile nodes that connect to the IVAN. The peer in the IVAN acted as a NAP (Network Access Point). The latter node was a laptop used as a bridge connecting the internal Ethernet with the Bluetooth network. The software used to implement the bridging between these two technologies is [<http://bridge.sourceforge.net/>].

9.3.1 Bluetooth Hardware for BlueZ

The USB devices used were supported by BlueZ out-of the box via the hot plug subsystem of the Linux distribution used. The UART device in the iPAQ was controlled via the hciattach command (e.g hciattach /dev/ttySB0 bcspp 230400). The local demonstrator at University of Bonn, also used PCMCIA device. For details about their configuration [<http://www.holtmann.org/linux/bluetooth/bt3c.html>] is beneficial.

9.3.2 Creating a software bridge

The software bridge was created via the following script.

```
brctl addbr nap0           # add bridge device nap0
brctl setfd nap0 0        # set forwarding delay to zero
brctl stp nap0 off       # disable spanning tree protocol (see note below)
ifconfig <ethX> 0.0.0.0 down # turn the dedicated Ethernet device down
brctl addif nap0 <ethX>  # add Ethernet interface to the bridge device
ifconfig nap0 <IP conf>  # configure nap0 with your <ethX> parameters
route add default gw <gw> # set your default gateway
ifconfig nap0 up         # enable your bridge device
ifconfig <ethX> up       # bring up the associated Ethernet device
```

9.3.3 Prepare for upcoming PAN connections

The creation of the bridge does not include the dynamic join of mobile nodes that are attaching to the IVAN. This can be solved by the following script that has to be placed in /etc/bluetooth/pan with the name dev-up and execution rights.

```
#!/bin/sh
# 'dev-up' script to do dynamic bridge management on GN
# $1 is the new if name, passed by 'pand'
brctl addif nap0 $1
ifconfig $1 0.0.0.0 up
```

9.3.4 Enabling and connecting to a NAP

The PAN service on the between the participating nodes is started via the pand commnd.. Here, the NAP ist started via the following command.

```
pand --master --listen --role NAP
```

When the service is running, the same command can be used on the client side with different parameter, where <BT Address> is the Bluetooth MAC address of NAP peer.

```
pand -connect <BT Address>
```

The Ethernet device name, which is created after a successful connection) can be chosen when executing pand. If the prefix bnep should be replace for example by eth then specify it via the -i option, e.g.:

```
pand -connect <BT address> -i eth2
```

9.4 WLAN

In addition to provide external connectivity, WLAN is also used as an access system inside the IVAN. For this purpose, we place an WLAN access point inside the car, configured to have the ESSID IVAN. Having different ESSIDs for all access points, it is always possible to distinguish between them and to specifically select a certain access network (e.g. to distinguish between external hot spots and the in-vehicular access point). The access point can be configured by using the internal Web-interface.

10 Conclusion

The OverDRiVE project successfully developed several demonstrators to validate key aspects of workpackage 2 (mobile multicast) and workpackage 3 (mobile router and IVAN management). The development followed evolutionary steps towards an overall OverDRiVE demonstrator which was shown at the HyWiN 2003 audit and the annual project review 2003 in Torino, Italy. The demonstration [http://dbs.cordis.lu/cordis-cgi/srchidadb?ACTION=D&SESSION=76702004-2-12&DOC=3&TBL=EN_RTDN&RCN=EN_RCN_ID:1633&CALLER=CORDISwire] utilized UMTS, GPRS, WLAN and DVB/T to deliver IPv6 uni- and multicast traffic to vehicles by using a mobile router. Dedicated applications like remote software download, remote access to the vehicle and (adaptive) uni- and multicast video streaming showed the potential application range for the OverDRiVE concept.

Besides the overall demonstrator, basically 2 other demonstrators were developed to highlight specific parts of the OverDRiVE work. The combination of micro- und macro-mobility approaches was used to optimize handover in large moving networks such as trains, ships, etc. The group management demonstrator for mobile multicast traffic showed the feasibility of the approach to optimize the radio and systems utilization for dynamic multicast groups using different access systems.

The demonstrations provided valuable input to the project in order to validate the concepts and to identify further working areas. Showing the demonstrators at several events increased the overall awareness of the project in the scientific community.

References

- [1] OverDRiVE Deliverable D03, “OverDRiVE Scenarios, Services, and Requirements”, September 2002, <http://www.ist-OverDRiVE.org>
- [2] OverDRiVE Deliverable D04, “Current Approaches to IP Multicast in a Mobile Environment”, November 2002, <http://www.ist-OverDRiVE.org>
- [3] OverDRiVE Deliverable D07, “Concept of Mobile Router and Dynamic IVAN Management”, March 2003, <http://www.ist-OverDRiVE.org>
- [4] OverDRiVE Deliverable D09, “Concepts for Mobile Multicast in Hybrid Networks”, May 2003, <http://www.ist-OverDRiVE.org>
- [5] OverDRiVE Deliverable D16, “Functional Description and Validation of the Mobile Multicast Architecture and the Group Management”, March 2004, <http://www.ist-OverDRiVE.org>
- [6] D. Estrin et al., “Protocol Independent Multicast-Sparse Mode (PIM-SM) Protocol Specification”, RFC2362, June 1998.
- [7] S. Deering, W. Fenner, and B. Haberman, “Multicast Listener Discovery (MLD) for IPv6”, RFC 2710, October 1999.
- [8] B. Fenner, H. He, B. Haberman H. Sandick, “IGMP/MLD-based Multicast Forwarding (“IGMP/MLD Proxying”)", draft-ietf-magma-igmp-proxy-04.txt, work in progress, September 2003.
- [9] V. Devarapalli, R. Wakikawa, A. Petrescu, P. Thubert, “Nemo Basic Support Protocol”, draft-ietf-nemo-basic-support-02.txt, work in progress, December 2003.
- [10] PIM-SM for IPv6 in FreeBSD, pim6sd, <http://www.freshports.org/net/pim6sd/>
- [11] VideoLAN VLC Media Player, <http://www.videolan.org/vlc/>
- [12] LIVSIX home page, <http://www.nal.motlabs.com/livsix>
- [13] HyWIN 2003, International Workshop on Hybrid Wireless Networks, Turin, 2nd December 2003, <http://www.ist-OverDRiVE.org/HyWiN2003>.
- [14] ETSI EN 301 192, " Digital Video Broadcasting (DVB);DVB specification for data broadcasting", V1.2.1, June 1999
- [15] ETSI EN 101 202, Digital Video Broadcasting (DVB);Implementation guidelines for Data Broadcasting ", V1.1.1, September 2000
- [16] R. Gilligan, E. Nordmark, “Transition Mechanisms for IPv6 Hosts and Routers“, RFC1933, 1996
- [17] VTun homepage: <http://vtun.sourceforge.net>
- [18] M. Handley, V. Jacobson, “SDP: Session Description Protocol”, RFC2327, April 1998
- [19] S. Olson, G. Camarillo, A. B. Roach, “Support for IPv6 in Session Description Protocol (SDP)”, RFC3266, June 2002] and switch via RTSP
- [20] H. Schulzrinne, A. Rao, R. Lanphier. “Real Time Streaming Protocol (RTSP)”, IETF RFC2326, April 1998

- [21] W. R. Stevens, B. Fenner, A. M. Rudoff, “UNIX Network Programming, The Sockets Networking”, Addison-Wesley Professional

Annex A Software Development for Linux-based Handhelds

A.1 Introduction

One goal of the OverDRiVE demonstrator is to show the integration of state-of the art handheld devices in mobile environments. This kind of devices often provide particular services to the user, i.e. they are serving as phone, personal information manages (PIM), gaming platform or media player. These differentiations may become obsolete as hardware advances and costs for the user declines while providing the performance for most handheld specific applications in one device.

During the project Linux was chosen as an important platform for the development and realization of the OverDRiVE demonstrator. One of the major benefits of Linux is the availability of open sources that can be used as a basic platform for further development in the OverDRiVE project.

Linux and handheld devices were chosen at an early stage in the demonstrator task. Here, we share information about Linux software development for handheld devices that was gained by different partners and assembled to a cook book for particular development tasks.

A.2 Cross-Compiling

- Many useful information can be found at <http://www.handhelds.org/> .
- Choose a compiler toolchain from <http://handhelds.org/download/toolchain/> , e.g. arm-linux-gcc-3.3.1-030818.tar.gz
- Install it to the default directory Put the cross compile tools in your path variable, e.g export PATH=/usr/local/arm/3.2.3/bin:\$PATH

A.2.1 ipkg-utils and ipkg

Ipkg-utils is needed for creation of kernel packages, ipkg is needed to create a modified boot image.

- Get the ipkg-utils from handhelds
 - cvs -d :pserver:anoncvs@cvs.handhelds.org:/cvs login (passwd is anoncvs)
 - cvs -d :pserver:anoncvs@cvs.handhelds.org:/cvs co ipkg-utils
- Get the ipkg sources from handhelds
 - cvs -d :pserver:anoncvs@cvs.handhelds.org:/cvs login (passwd is anoncvs)
 - cvs -d :pserver:anoncvs@cvs.handhelds.org:/cvs co ipkg

Info: The script may need a particular automake and autoconf version, we use automake-1.5 and autoconf2.57

A.3 Cross-Compiling the Linux kernel

- Get the kernel sources from handhelds
 - `cvs -d :pserver:anoncvs@cvs.handhelds.org:/cvs login` (passwd is anoncvs)
 - `cvs -d :pserver:anoncvs@cvs.handhelds.org:/cvs co linux/kernel`

Kernel versions are available at handhelds.org: The kernel for the iPAQ is constantly evolving and there is not always a stable version in the head branch of the mentioned CVS. If the current kernel version is not stable or there is no need to test the latest version then the checkout procedure should specify a dedicated release. By time of writing the latest release that seems to be stable enough for the iPAQ model H3800 is 2.4.19-rmk6-pxa1-hh23. This release can be checked out by the following command:

```
cvs -d :pserver:anoncvs@cvs.handhelds.org:/cvs
  co linux/kernel -r K2-4-19-rmk6-pxa1-hh23 linux/kernel
```

For the iPAQ H3800 (for other machines take a look at README.HANDHELDS) perform the following steps:

```
make ipaqsa_config
```

```
make menuconfig (and disable mwvlan wireless LAN driver)
```

```
Livsix: disable IPv6 and patch slip.c, dev.c, netsyms.c
```

```
It's a good idea to edit the EXTRAVERSION in the kernel Makefile,
e.g. EXTRAVERSION = -rmk6-pxa1-hh23-livsix
```

```
make oldconfig && make dep && make zImage && make modules
```

A.4 Create Kernel Packages

```
Create directory and go in, e.g. mkdir packages && cd packages.
```

```
Call script from kernel directory:
```

```
<path to kernel src>/scripts/ipkg-make-kernel-packages /<absolute path to kernel src>
```

```
Info: The script call ipkg-build, which is a script from the ipkg-utils package
```

A.5 Compile Sound Module

- Get the alsa modules:
 - `cvs -d :pserver:anoncvs@cvs.handhelds.org:/cvs co alsa`

```
go to the alsa-driver directory
```

```
./config.ipaq <absolute path to the compiled kernel sources>
```

```
Info: Don't care about the fatal error that the configure script prints
```

```
make
```

```
as root perform: make ipkg
```

```
cp familiar/alsa-modules_<version>.ipk ../packages/
```

Finalize Package Update

Got to package dir

```
ipkg-make-index . > Packages
```

Now, copy all files in the package dir to the ftp server and update the kernel or create a new boot image for the iPAQ (cf A.6).

A.6 Images for the iPAQ

Your kernel must support mtdcore, mtdram, mtdblock, jffs2!

A.6.1 Mounting an jffs2 image

Get an image from handhelds.org, e.g. bootgpe2-v0.7.1-h3600.tar

Untar it (we need bootgpe2-v0.7.1-h3600.jffs2)

```
insmod mtdcore
```

```
insmod mtdram total_size=32768 erase_size=256
```

```
insmod mtdblock
```

Info: Now you should have an Memory Technology Device (dev/mtdblock/x, x=0,1,2,...)

Copy the image, e.g. `dd if= bootgpe2-v0.7.1-h3600.jffs2 of= dev/mtdblock/0`

Mount the image, e.g. `mount -t jffs2 /dev/mtdblock/0 /mnt/jffs`

A.6.2 Modify an iPAQ image

If you just want to provide files, go ahead and copy them to the mounted image. If you want to upgrade packages compile ipkg :

Create a local ipkg.conf file:

```
src kernel ...
```

```
ipkg -f <path to local ipkg.conf>/ipkg.conf -o /mnt/jffs update
```

```
ipkg -f <path to local ipkg.conf>/ipkg.conf -o /mnt/jffs upgrade
```

Example ipkg.conf file for offline operation:

```
src kernel file://home/pilz/work/ipaq/packages
```

```
dest root /
```

```
dest ram /mnt/ramfs
```

```
dest ext /mnt/hda
```

Attention: The postinstall scripts are not executed if you run ipkg in offline mode (option `-o`). Therefore some things must be done by yourself.

Example: After upgrading the kernel, the link `/boot/zImage` should be checked.

To creating the image use:

```
mkfs.jffs2 -o test.jffs2 -d /mnt/jffs2 -p -e 0x40000
```

A.6.3 Booting the Image

Attention: All data on your iPAQ will be lost!

Here, we only describe the installation via a CF Card.

Copy test.jffs2 to an CF card

Insert CF card into jacket, reboot

Go to bootldr prompt and type:

sleeve insert

pcmcia insert

vfat mount 0

copy hda1:test.jffs2 root

boot

Now the new image is loaded and all your data is gone...

Have fun.

A.7 Native compiling

An alternative to cross-compiling is the native compilation on the mobile device itself. As these devices do offer performance comparable with modern desktop computers and because of the limited amount of storage do not have a full Linux distribution installed cross-compiling is the better solution if it can be applied. However for an application which has many dependencies to other libraries, e.g. a video streaming client, it may be more appropriate to compile the application natively.

The intimate project [<http://intimate.handhelds.org/>] offers a full debian based Linux distribution for ARM based handheld PCs like iPaqs. It is based on the familiar Linux distribution and additionally supports the debian packet management. Minimum requirements for the base image are currently approximately 140MB of storage. Therefore a storage extension card is necessary for the installation. A microdrive is suggested by the developers and was used in the OverDRiVE project.

A.8 Example: Cross-Compiling Livsix for the iPAQ

Get Livsix:

```
cvs -d :pserver:guest@cvs.nal.motlabs.com:/cvsroot login
cvs -d :pserver:guest@cvs.nal.motlabs.com:/cvsroot co
```

Alternative: Compile Livsix module on an iPAQ that runs the Intimate distribution. This requires an iPAQ that is dedicated to that Linux distribution and a big CF card.

Compiling Livsix with a cross compiler:

run the WRAPUP script

modify linux-gnu-kernel/Makefile.in:

```
livsix_o_LDFLAGS: change elf_i386 to armelf
INCLUDES: replace /usr/src/linux (use headers form kernel compilation)
CFLAGS: replace /usr/src/linux (use headers form kernel compilation)
LINK=arm-linux-ld
```

run

```
CC=/usr/local/arm/3.3.1/bin/arm-linux-gcc ./configure --enable-debug --host=arm-linux
```

A.9 Example: Compiling mpeg4ip on a intimate Linux iPAQ

Versions 0.9.8 and 0.9.9 of the mpeg4ip player were compiled on the iPAQ.

Mpeg4ip is configured via a script `./bootstrap`. This script calls `./configure` of all sub packages. Because of the limited storage capabilities of the iPAQ the following shell environment variables have to be set before calling `./bootstrap` in order to prevent the compilation with debug information. This is because the default compiler option is set to “-g”. With the installed configuration of the iPAQ the compilation process aborted with several warnings. They were ignored by removing the `-Werror` compiler option of the regarding Makefiles.

The resulting binaries were not able to load the XVID [<http://www.xvid.org/>] video decoder plugin. However, the alternative mpeg4ip decoder plugin `mpeg4_iso_plugin` had severe performance problems on the iPAQ, probably due to usage of floating point operations which are not supported in hardware and are therefore very slow on arm based architectures. An alternative, faster floating point operation emulation module did not bring a significant improvement in video decoding performance. An mpeg4 video could not be displayed. However after patching the xvid codec – the `emms` instruction is not supported on the iPAQ – the mpeg4ip player was able to load the xvid module which runs with enough performance on the iPAQ.

The audio codecs included in mpeg4ip had the same performance problems with the arm architecture. For this reason a plugin was developed which uses the mad mp3 audio codec [sourceforge.net/projects/mad/] which uses fix point arithmetic and mostly avoids floating point operations.

Annex B Software Update - Setup and Configuration

B.1 Installation guide

These are the installation instructions for Update application. For convenience it also covers the basic installation of the *JAVA SDK*, the *Jakarta Tomcat* JSP/Servlet container and the build tool *ant*. Skip the appropriate steps if that is already running on the target machine. Further on it is assumed that the target machine is an i386 platform running Microsoft Windows 2000 or XP with basic TCP/IP services installed.

B.1.1 Java SDK installation

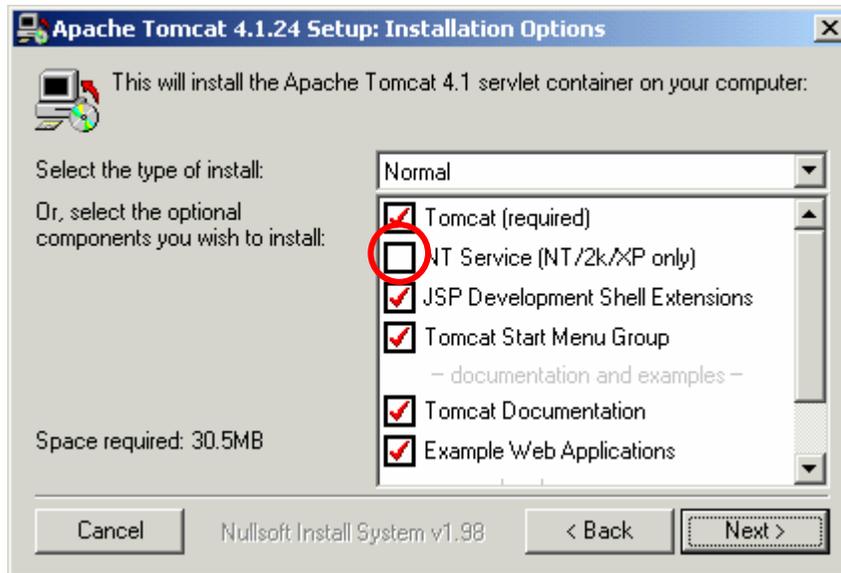
- Download a Java Development Kit (JDK) release (version 1.4 or later) from: <http://java.sun.com/j2se/> or use the JDK release on the installation cd.
- Install the JDK according to the instructions included with the release.
- Set an environment variable `JAVA_HOME` to the pathname of the directory into which you installed the JDK release.

B.1.2 Tomcat 4.1.24 installation

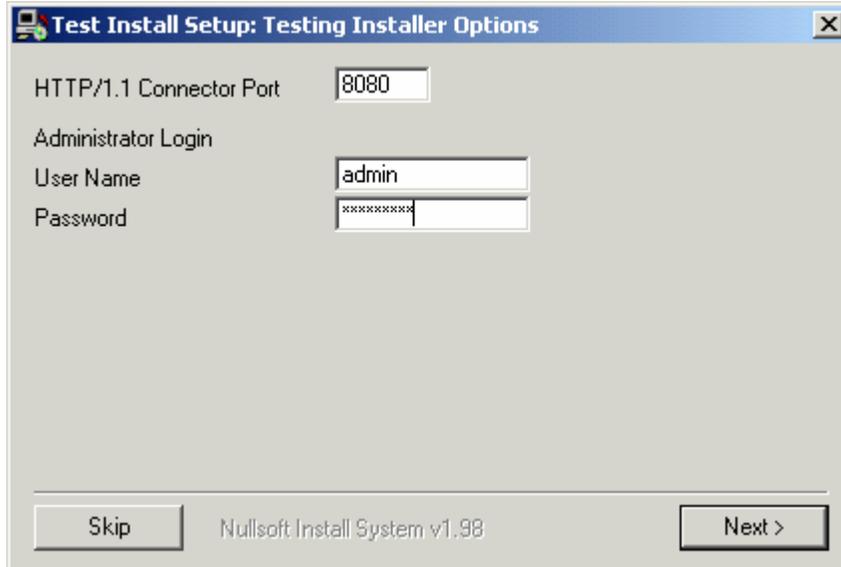
The Update Server Application is a Java Web Application and as such requires a JSP/Servlet container to run. Currently the only supported JSP/Servlet container is Jakarta Tomcat 4.1

To install Jakarta Tomcat 4.1 run *jakarta-tomcat-4.1.24.exe* and follow the installation instructions

If you want to install Jakarta Tomcat as NT Service, you have to choose it here:



To complete the installation set the port on which Jakarta Tomcat listens, the administrator login and password. This login and password is required later for the installation of the update server application.



Finally set an environment variable TOMCAT_HOME to the pathname of the directory into which you installed Jakarta Tomcat.

B.1.3 ant installation

- Unzip the *jakarta-ant-1.5.1.zip* archive.
- Set an environment variable ANT_HOME to the pathname of the directory into which you unzip the jakarta-ant-1.5.1 archive.
- Add %ANT_HOME%\bin to your PATH environment variable.

B.1.4 Update server and Web-interface installation

- Unzip the “*Remote Update v0.9.zip*” archive.
- Copy all *.jar files from the <Remote Update v0.9>\UpdateServer\lib directory and the *client.jar* file from the <Remote Update v0.9>/Client-Web-interface/lib directory into the <TOMCAT_HOME>\common\lib directory.
- Go to <Apache_Home>\htdocs and make a directory named dtd. Copy all *.dtd files from the <Remote Update v0.9>\dtd directory into the <Apache_Home>\htdocs\dtd directory. If you have no Apache server running on your system, change the path of the DTD in *update-template.tld*, which you find in <Remote_Update v0.9>\Client-Web-Interface\web, to <Remote_Update v0.9>\dtd.
- In a terminal window go to the directory, where you unzip the Remote Update archive. For example:
`cd “c:\Remote Update v0.9”`
- Open the *build.properties* file to setup the general *ant* build options for the UpdateServer and Client-Web-Interface application.

Set the following options:

| | | |
|-------------------|---|---|
| username | - | Adminstrator login of the tomcat server. |
| password | - | Adminstrator password of the tomcat server. |
| tomcat-url | - | URL of the Jakarta Tomcat server. |

- Go to the *UpdateServer* directory and open the *build.properties* file for the UpdateServer.
Set the following options:

| | | |
|---------------------|---|---|
| project-root | - | root directory of the UpdateServer for example „c:/Remote Update v0.9/UpdateServer/“ |
| project-name | - | Application name. |

Note: All paths have to be separated with “/” (slash) and not with “\” (backslash)!

- Go to the Client-Web-Interface directory and open the *build.properties* for the Client-Web-Interface. Here you have to set the values for **project-root** and **project-name** as well.

- If Tomcat is not running, start Tomcat by run *startup.bat* from the <TOMCAT_HOME>\bin directory. Wait until Tomcat is started!
- Go to the <Remote Update v0.9> directory and run *build.bat*. This task compiles the source, packages the *.war* file and deploys the applications into Tomcat.

If you want to install only the Update server or the Client-Web-Interface, go to the <Remote Update v0.9>\UpdateServer or <Remote Update v0.9>\Client-Web-Interface directory and run *ant deploy*.

- The UpdateServer and Client-Web-Interface applications should be installed now

B.2 Configuration guide

B.2.1 The UpdateServer configuration guide

B.2.1.1 Server properties(*server.properties*)

The UpdateServer is configured with the file *server.properties*, which has to be in the directory <TOMCAT_HOME>\webapps\UpdateServer\WEB-INF.

There are only two parameters in this file: *UpdateList* and *UpdateLocation*. The first value is the full and filename of the update list file and the second value is the path of the update files.

B.2.1.2 The update list file

After the installation the update list file has the filename *updates.xml* and is in the directory

<TOMCAT_HOME>\webapps\UpdateServer\WEB-INF

When you want to change the location or generate a new update list file, you have to change the *UpdateList* property in the file *server.properties*.

The update list file contains the information about all updates available on the server. It has an xml structure, which consists of a list of update object. Every update element contains the child elements *version*, *target_systems*, *files* and *info*.

The *version* element contains the version of the update element.

The *target_systems* element contains a list of *system* elements. Every *system* element has an *id* attribute and the child elements *model_nr* and *model_name*.

The *files* element contains the child elements *file_count* and *file*. Every *file* element has two child elements *name* and *type*. The *name* element contains the filename and the *type* element contains the filetype.

The *info* element contains the general description of the update element.

This is a small example of an update list file:

```
<updates>
  <update id="0001">
    <version>1.0</version>
    <target_systems>
      <system id="OverDRiVE">
        <model_nr>332</model_nr>
        <model_name>smart</model_name>
      </system>
    </target_system>
    <files>
      <file_count>1</file_count>
      <file>
        <name>test.xml</name>
        <type>xml</type>
      </file>
    </files>
    <info>This is a small sample update</info>
  </update>
</updates>
```

B.2.2 The Client-Web-Interface configuration guide

B.2.2.1 The client properties(*client.properties*)

The Client-Web-Interface is configured with the *client.properties* file, which has to be in the directory `<TOMCAT_HOME>\webapps\Client-Web-Interface\WEB-INF`.

Here is a description of each parameter:

- *system_id*: Name of the system for which the client should request the update server for updates. For example *Modis* or *OverDRiVE*.
- *model_nr*: Car model number for which the client should request the update server for updates. For example: *221*
- *model_name*: Car model name for which the client should request the update server for updates. For example: *E-Klasse*
- *UpdateServerURL*: Url of the update server.
- *UpdateListServletURL*: Relative url of the servlet which returns the list of updates available for the system. For example: */UpdateServer/getUpdateList*
- *UpdateServletURL*: Relative url of the servlet which returns the update specified in the request. For example: */UpdateServer/getUpdate*
- *TargetSiteURL*: Url of the site, to which you want to go after the update process has finished.
- *packages*: Number of packages in which the message should be divided before sending. **Not implemented yet**
- *minSize*: **Not implemented yet !**
- *maxSize*: **Not implemented yet !**
- *saveUpdateListTo*: Absolut path of the directory, where you want to save the *updatelist.xml* file, which is received from the update server und read by the client to display the list of available updates.

- *saveUpdateTo*: Absolut path of the directory, where you want to save the files, which will be send as attachment with the update response message.
- *ipVersion*: Possible values are *v4* and *v6*. **Not implemented yet !**

Note: All pathes have to be seperated with „/“(slash) and not with „\“(backslash) !!!

3. Typical Sequence of Operation
- 3.1. Operation without Back-Channel
- 3.2. Operation with Back-Channel
4. Security Considerations
5. References
6. Authors' Addresses

1. Overview of the Protocol

The BTFTP protocol was created for the purpose of data and file transfer over digital broadcast channels, DVB like ([DVB-S], [DVB-T]). The peculiar features of the protocol are:

- a) optional back-channel: the protocol works without back-channel, in a pure broadcasting way; an optional back-channel is nevertheless supported to increase reliability
- b) very small header overhead: the headers are kept as small as possible
- c) simplicity: the specification and implementation are very simple (more straightforward than DVB Object Carousel for example)
- d) extensibility: the BTFTPWriteRequest packet can easily be improved for particular needs.
- e) compatibility: is an IP based protocol (instead of DVB Data Carousel and Object Carousel Protocols)

The protocol was thought to be implemented over User Datagram Protocol (UDP) but this does not exclude other datagram protocols. In that way it can be encapsulated in Digital Video Broadcasting transport stream (Multi-Protocol Encapsulation Profile [ISO13818]) and it can be sent over wireless, broadcasting channels. BTFTP can work in two ways: without the back-channel and with a low bit-rate back-channel.

- Operation without back-channel

The protocol can be used without a back-channel, in a pure broadcasting manner, with no interactivity. This is the main purpose of the protocol.

In that way the contents are fixed, and the user selects a channel and then begins downloading all or part of the data of that channel.

Reliability is obtained in two ways: with the Forward Error Correction (FEC) of the underlying layers (e.g. Viterbi and Reed Solomon protection of a DVB channel) and/or redundancy, i.e. repeating the transmission.

- Operation with back-channel

The protocol can be used with a low bit-rate back-channel. This adds reliability at the expense of another channel that should carry the packet requests. The protocol in itself does not implement interactivity (it was not designed for this scenario): for this purpose HTTP or other protocols can be possibly used instead of BTFTP. The check of data integrity is provided in both scenarios by a CRC32 code.

2. Protocol Specification

This section describes the format of the BTFTP packets. Below is shown the syntax of each type of packet and the semantic of each field.

For UDP encapsulation, only one BTFTP packet MUST be encapsulated in one UDP packet. In the same way, a UDP packet MUST contain only one BTFTP packet.

It should be noted that the protocol does not specify the order for packet transmission; for transmission the order specified by BlockNumber SHOULD be used. Packets can be duplicated (e.g. to increase reliability).

2.1. BTFTPPacket

This is the primary packet which contains the common header of all BTFTP packets. The packet format is:

```
BTFTPPacket()          Length in bits (big endian notation)
{
TID;                   32
OPCODE;                8
BlockNumber;           32
BlockSize;             16
FutureUse;             5
LastPacketFlag;       1
RedundancyFlags;      2
if (OPCODE==0)
    BTFTPWriteRequest();
if (OPCODE==1)
    BTFTPData();
if (OPCODE==2)
    BTFTPNak();
if (OPCODE==3)
    BTFTPWriteRequestExtension();
if (RedundancyFlags==01)
```

```
        Chekcsun;                32
If (RedundancyFlags==10)
        CRC32;                    32
If (RedundancyFlags==00)
        Ignored;                  32
}
```

Semantics:

TID: it is the Transmission Identifier which is used to differentiate each file transmission from others, in this way we can have different multiplexed file transmissions. The value of this field is the same for each packet of the same file transmission, and must be different from other file transmissions at the same time.

OPCODE: this field contains the operation code of current packet, this value specifies the packet type which can be BTFTPWriteRequest (value = 0), BTFTPData (value = 1), BTFTPNak (value = 2) or BTFTPWriteRequestExtension (value = 3).

BlockNumber: this is the packet counter of the protocol, it is used to enumerate packets of same type; the start value is 1 and it's incremented by 1 at each new packet. E.g. BlockNumber = 1 for the first packet of BTFTPData, BlockNumber = 2 for the second packet of BTFTPData and so on.

BlockSize: It is the size in octets of the current packet. There are no restrictions in the use of this field but it MUST be used a constant BlockSize value for BTFTPData packets for each single file transmission; only the BlockSize of the last packet of BTFTPData can be shorter.

FutureUse: five bits field for future use.

LastPacketFlag: this field is used only with OPCODE = 1 (BTFTPData), the value 1 indicates that this is the last data packet of a trasmitted file. If OPCODE is not BTFTPData the field is ignored.

RedundancyFlags: these two bits specify the redundancy type of the BTFTP packet.

Checksum: this is a 32 bits field which contain the sum of all octets in the packet without carry.

CRC32: this is an 32 bits field which contain the CRC calculated over this packet, which must be calculated in according to MPEG 2 system specifications, refer to [ISO13818]

Ignored: if RedundancyFlags is 00 these 32 bits are ignored by the decoding procedure.

2.2. BTFTPWriteRequest (OPCODE = 0)

The Write Request Packet is used to inform the receiver that a

transmission is beginning, this packet carries the information necessary to file decoding, it contains information like file name, file type or generic file information.

All information in the Write Request part of packet are encapsulated as TLV (Type Length Value) format. TLV format is described below.

There is only one Write Request packet for each file transmission (same TID), if the packet is too short to encapsulate file information or if there is the necessity to transmit extra file description it must be done using the BTFTPWriteRequestExtension.

The format of the Write Request is:

```
BTFTPWriteRequest()          Length in bits (big endian notation)
{
    TotalBlockNumberWRE;      32
    TotalBlockNumberData;    32
    TLV();
}
```

Semantics:

TotalBlockNumberWRE: this field carries to the receiver the total block number of Write Request Extension packets. If the value is 0 there are no Extension packets. Otherwise the value specifies how many Write Request Extension packets the receiver must read to completely rebuild data description.

TotalBlockNumberData: this field indicates the number of packets containing data, the value specifies how many BTFTPData packets the receiver has to read to rebuild the transmitted file.

TLV(): indicates a sequential list of TLV fields which contains the data description, the file name TLV is mandatory. This TLV list terminates with a type '0' TLV. Look at “Types Used in TLV fields” section for more information.

The general form of a TLV() is:

<Type><Length><Value, i.e. content (when specified can be omitted)>

2.3. BTFTPData (OPCODE =1)

This is the data packet of the protocol. A portion of the file to transmit is stored in the payload. The length of payload depends on the BlockSize value and on the extra information optionally encapsulated.

The format of this packet is:

```
BTFTPData()
{
    TLV();
}
```

```
    Payload;  
}
```

Semantics:

TLV(): indicates an optional sequential list of TLV field which contains the extra data description, the type '0' TLV is mandatory. This TLV list must terminate with a type '0' TLV. See TLV section for more information.

Payload: it contains a portion of transmitted file, the size depends on the BlockSize value and the length of previous TLV list (look at the meaning of BlockSize).

2.4. BTFTPNak (OPCODE = 2)

This type of packet is used only in operation with back channel. BTFTPNak packets are sent by the receiver to the transmitter. BTFTPNak packets must be sent with the same TID of the requested packet and must be single packets (the BTFTPNak cannot be on multiple packets). For example, if packet n of a transmission with TID m has been lost, the BTFTPNak request is a packet with the same TID m. BTFTPNak packets contain requests of lost packets.

The format of this packet is:

```
BTFTPNak()  
{  
    TLV();  
}
```

Semantics:

TLV(): this is one Nak TLV field followed by a type '0' TLV. The Nak TLV indicate the list of lost packets.

The TLV with the request(s) is of type 128: it contains a list of intervals of packets to be retransmitted (Nak list). Each entry of the list is 9 octets long. The format of the list is:

<OPCODE, 1 octet><Block number of first packet, 4 octets><Number of packets, 4 octets>

<OPCODE><Block number of first packet><Number of packets> ...

The notation is big endian. For example, if data packet 0x0A31 (hexadecimal) had been lost, the TLV content would be:

```
0x01 0x00 0x00 0x0A 0x31 0x00 0x00 0x00 0x01
```

If there were 0x39 lost packet from packet number 0x0A31, and 0x28 from data packet number 0xBA13, the content of the TLV would be:

```
0x01 0x00 0x00 0x0A 0x31 0x00 0x00 0x00 0x39
```

```
0x01 0x00 0x00 0xBA 0x13 0x00 0x00 0x00 0x28
```

2.5. BTFTPWriteRequestExtension (OPCODE = 3)

This type of packet is optional and is used only if one BTFTPWriteRequest packet can not carry all the file and transmission description (because the description length is greater than the payload size of the BTFTPWriteRequest packet).

The format is:

```
BTFTPWriteRequestExtension()  
{  
    TLV();  
}
```

Semantics:

TLV(): it contains a list of TLV field following by a type '0' TLV.

2.6. Types used in TLV fields

TLV fields have a coherent convention for the Type field. Type field is 1 octet long. In general, the number of octets of the Length field depends on the Type field; in other words Length field is not fixed for all Types.

It should be noted that all TLV must be included in a specific packet, except for TLV of Type 0. Type 0 TLV MUST be included whenever a TLV list is specified by the protocol, to indicate that the list is terminated and it must be the last TLV (the only TLV for empty lists).

The Type field has to be specified with the following convention:

Type = 0: There are no Length and Value fields.

Type = 1: Length of 2 octets. The Value contains the file name.

Type = 2: Length of 2 octets. The Value contains the type of the file.

Type = 3: Length of 2 octets. The Value contains text information about the file.

Type = 4: Length of 2 octets. The Value contains a command file, to be executed on a local machine.

Type = 5: Length of 1 octets. The Value contains the actual length of the file.

Type = 128: Length of 2 octets. The Value contains a Nak list.

In detail we have:

Type = 0: This Type tells the decoder that the TLV sequence has come to an end. It MUST be the last TLV of the list and it MUST be present in every TLV list.

Type = 1: The name of the file that is going to be transmitted. The filename is encoded in standard 8 bits ASCII code. It MUST be present if a file is transmitted. It MUST be included (if present) only in BTFTPWriteRequest or BTFTPWriteRequestExtension packets.

Type = 2: The type of the file. The possible types (e.g. gzip, text, html, png...) are not standardized in this document. The type is encoded in standard 8 bits ASCII code. It MUST be included (if present) only in BTFTPWriteRequest or BTFTPWriteRequestExtension packets.

Type = 3: Some useful information on the file. This can be an ASCII string that describes the file. It is, like the other TLV, optional. The information is encoded using standard 8 bits ASCII code. It MUST be included (if present) only in BTFTPWriteRequest or BTFTPWriteRequestExtension packets.

Type = 4: Sometimes it is useful to have a command file to be executed at application level and to be transmitted very rapidly, possibly on a single packet, which is the purpose of this TLV. This should be a text file. It MUST be included (if present) only in BTFTPWriteRequest or BTFTPWriteRequestExtension packets.

Type = 5: Sometimes it is useful to know the exact length of the file in the BTFTPWriteRequest. When this TLV is present, the last packet of BTFTPData can be of the same length of other BTFTPData packets. It MUST be included (if present) only in BTFTPWriteRequest or BTFTPWriteRequestExtension packets.

Type = 128: It contains a list of intervals of packets to be retransmitted (Nak list). Each entry of the list is 9 octets long. It MUST be included (if present) only in BTFTPNack packets.

The format of the list of intervals is:

```
<OPCODE, 1 octet><Block number of first packet, 4 octets>
<Number of packets, 4 octets>
<OPCODE><Block number of first packet><Number of packets> ...
```

Other values for the Type field are reserved for future use.

3. Typical Sequence of Operation

This chapter describes two typical sequences of operation and it should clarify the previously stated concepts. Note that this document is not intended to specify a policy for requests (time-out algorithm, request reiteration, et al.)

3.1. Operation without Back-Channel

In a typical scenario without back-channel, let's suppose the Receiver has begun listening to a channel (address and port). The Broadcaster begins transmitting a file:

Transmitter Receiver

BTFTPWriteRequest Receiving BTFTPWriteRequest, checking CRC32

BTFTPData Receiving BTFTPData, checking CRC32
and sequence number of the packet.

The packet is out of sequence, set a timeout for previous packets.

The packet is in sequence, but CRC32 is wrong: that TID is discarded.

The timeout for that TID is reached: that TID is discarded.

The packet is in sequence and CRC32 is right: continue until the last packet is correctly received.

3.2. Operation with Back-Channel

In a typical scenario with back-channel, let's suppose the Receiver has begun listening to a channel (address and port). There is also a channel for request packets, that can be chosen as a low bit-rate channel. The Broadcaster begins transmitting a file:

Transmitter Receiver

BTFTPWriteRequest Receiving BTFTPWriteRequest, checking CRC32

BTFTPData Receiving BTFTPData, checking CRC32
and sequence number of the packet.

If the packet is out of sequence, set a timeout for previous packets.

If the packet is in sequence, but CRC32 is wrong: add to the list of packets to be requested. Periodically send a BTFTPAcknowledgement packet to request lost or wrong packets.

If timeout for that TID is reached: that TID is discarded.

If the packet is in sequence and CRC32 is right: continue until the last packet is correctly received.

A clever policy for lost packets requests and back-channel bandwidth minimization can be chosen during the implementation; this policy is not specified in this document.

4. Security Considerations

The protocol has to work without a back-channel (and it is the main purpose for which it has been developed). So a session can't be established in all scenarios. The content can be protected through a mechanism like this:

- the Receiver creates a private key and a public key pair
- the Receiver sends the public key to the Transmitter, out of band if there is no back-channel
- the Transmitter validates the key

- the Transmitter creates periodically symmetric keys (session keys). These keys are sent to the Receiver, encrypted with the Receiver's public key
- the Transmitter sends content (files) to the Receiver encrypting it with the session key

The protocol works on an UDP/IP stack, so a lot of security considerations concerning protocol on this stack can be applied to the BTFTP protocol.

5. References

[RFC1350] K. Sollins: "The TFTP Protocol - Revision 2", RFC1350, July 1992

[RFC2119] S. Bradner: "Key words for use in RFCs to Indicate Requirement Levels", RFC2119, March 1997

[ISO13818] "ISO/IEC 13818 Specification"

[DVB-S] EN 300 421: "Digital Video Broadcasting (DVB); DVB framing structure, channel coding and modulation for 11/12 GHz satellite services", ETSI, 1997

[DVB-T] EN 300 744: "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television", ETSI, 1999

6. Authors' Addresses

Paolo Casagrande
RAI - CRIT
Corso Giambone 68
10135 - Torino
ITALY

Luca Vignaroli
RAI - CRIT
Corso Giambone 68
10135 - Torino
ITALY

Annex D Responsibilities of Partners

Applications

| | |
|---|---------------------|
| Streaming | CRM, RAI, EED |
| Car Web-Server | DC |
| Software download | DC |
| CAN-Server and web-based access to CAN | DC |
| Web Access (IPv4/IPv6 Web access from car) | UBN |
| Software | |
| MR Livsix stack | CRM |
| Frontbox & gateway GPRS | EED, BUTE, CRM, RAI |
| Frontbox & gateway UMTS | CRM, RAI |
| Frontbox & gateway DVB/T | RAI |
| CAN-Server | DC |
| Web-Server | DC |
| VideoLAN, Darwin, Collaborator (client-SW, server-SW) | CRM, RAI |
| Multicast things, VIC/RAT (tested SW, know-how) | CRM,EED |
| Squid / wwwoffle proxy (totd/www6to4 software) | UBN |
| Group Management (network SW & client SW) | EED |
| BTFTP | RAI |
| Outgoing interface dispatcher for MR | UBN, DC |
| Hardware | |
| 2x PDA and 2x Laptops with WLAN and Bluetooth | UBN,CRM |
| In-car Bluetooth access point | DC |
| In-car CAN-Server and Web-Server | DC |
| Another LFN (if needed) | DC |
| External Web-Server | UBN, DC |
| External VideoLAN, Darwin Server | RAI |
| External Multicast, group management HW needed? | EED |
| Further external DVB-T HW needed | RAI |
| DVB-T Network | RAI |

Content

| | |
|---------------------------------------|---------------|
| SW, formats, content preparation etc. | EED, UBN, RAI |
|---------------------------------------|---------------|