TAMPERE UNIVERSITY OF TECHNOLOGY
Degree Programme in Information Technology

TERO KESKI-VALKAMA

CARBOOK: A PLATFORM FOR MOBILE AUTOMOTIVE SERVICES

Master of Science Thesis

Examiner: Professor Ilkka Haikala
Examiner and topic approved in
the Information Technology
Department Council
meeting on 4 June 2008

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
KESKI-VALKAMA, TERO
Carbook: A Platform for Mobile Automotive Services
Master of Science Thesis, 54 pages
July 2008
Major: Software Engineering
Examiner: Professor Ilkka Haikala
Keywords: Internet, car, SOA, Semantic Web

Wireless mobile technologies have triggered a rapid development of secondary network technologies. One such prominent field of technology is interoperability for consumer devices. This field is mostly based on XML and Web Services and it includes technologies such as Universal Plug-and-Play, open media container formats, open codecs and Rich Internet Application technologies for mobile devices.

Automotive field has been relatively slow and conservative in embracing these new Internet technologies. This is about to change as European Union and other substantial players are pressing forward with the safety and environmental technologies in cars. These technologies depend heavily on wireless Internet connectivity.

As part of this thesis work, I have played a central role in defining the core concept of a distributed framework for mobile automotive services, Carbook System. I have also outlined the first phase of a shared research environment, Carlab, for these kinds of services. Carlab is used to demonstrate different technologies in accordance to Elektrobit's vision for the future automotive Internet services. Carbook System will be implemented incrementally jointly with the continuation of the Carlab implementation.

In this master of science thesis I have mapped and evaluated the essential technologies and created a preliminary outline for Carbook System and a set of services. The first phase Carlab network topology and emulation of different domains in Carbook System are also drafted in this thesis work.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
KESKI-VALKAMA, TERO
Carbook: A Platform for Mobile Automotive Services
Diplomityö, 54 sivua
Heinäkuu 2008
Pääaine: Ohjelmistotuotanto
Tarkastaja: Professori Ilkka Haikala
Avainsanat: Internet, auto, SOA, Semantic Web

Langattomat mobiiliteknologiat ovat laukaisseet nopean kehityksen aallon seuraavan asteen verkkoteknologioissa. Yksi tällainen näkyvä teknologiakenttä on yhteensopivuus kuluttajalaitteiden välillä. Tämä kenttä rakentuu pääosin XML-notaation ja Web Services -teknologian päälle, ja sisältää teknologioita kuten Universal Plug-and-Play, avoimia mediasisältö -tiedostomuotoja, avoimia koodekkeja ja Rich Internet Application -teknologioita kohdennettuna mobiililaitteille.

Autoteollisuus on ollut suhteellisen hidas ja konservatiivinen näiden uusien Internet-teknologioiden käyttöönotossa. Tilanne on kuitenkin muuttumassa, kun Euroopan Unioni ja muut isot toimijat ovat ajamassa turvallisuuteen ja ympäristönsuojeluun liittyviä teknologioita autoihin. Nämä teknologiat riippuvat voimakkaasti langattomasta Internet-yhteydestä.

Tässä diplomityössä olen ollut keskeisessä roolissa määrittelemässä Carbook-järjestelmää, hajautettua kehysjärjestelmää mobiileille autopalveluille. Olen lisäksi suunnitellut ensimmäisen vaiheen jaetusta tutkimusympäristöstä, Carlabista, tällaisia palveluita varten. Carlabia käytetään eri tulevaisuuden auto- ja Internet-palveluihin liittyvien teknologioiden esittelemiseen Elektrobitin tulevaisuusvision ohjaamana. Carbook-järjestelmä toteutetaan askeleittain Carlabin toteuttamisen rinnalla.

Tässä diplomityössä olen kartoittanut ja arvioinut keskeiset teknologiat sekä luonut hahmotelman Carbook-järjestelmästä ja siihen liittyvistä palveluista. Myös Carlabin ensimmäisen vaiheen verkkotopologia ja eri Carbook-järjestelmän toimintakohteiden emulaatio on ääriviivailtu tässä diplomityössä.

PREFACE

I have written this master of science thesis at Elektrobit Oyj. This thesis is a part of a larger research effort towards ubiquitous Internet connectivity in cars.

I thank EB director Hannu Hakala for the massive amount of support and guidance he has given to this work. Thanks go also to professor Ilkka Haikala who has spent a lot of time to get me graduated. I also thank my colleagues and my fiancée Katja Ristilä for making this work possible.

Tampere, Finland, July 2008

Tero Keski-Valkama
Kärkikuja 2 B 38
33720 Tampere
tel. +358 (0)40 7069 762

CONTENTS

Terms and abbreviations

Term	Description
ADAS	Advanced Driver Assistance Systems
API	Application Programming Interface
B2B	Business-to-Business - A segment of eCommerce related to transactions and business processes divided between multiple participants.
Blog	Web-based log of human readable entries that can be personal or community based.
Bluetooth	A technology for short range wireless radio-link communication between small consumer devices.
CAN	Controller Area Network - One of the standardized ways of interconnecting vehicular electronic systems.
Carbook	A concept of the future automotive mobile Internet services.
Carlab	A physical laboratory implementation for testing automotive mobile technologies.
COSCAR	Context-Sensitive Mobile Services for Cars
DLNA	Digital Living Network Alliance - International collaboration of companies that aims to improve the interoperability for consumer media applications.

Term	Description
Domain	In this thesis, the word “domain” is used to describe different environments where the user is interfacing Carbook System or different execution environments for software. For example: Car domain, Home domain.
DVB	Digital Video Broadcasting - International open standard for digital television.
Ethernet	A family of network technologies for LAN connectivity.
FlexRay	One of the standardized ways of interconnecting vehicular electronic systems.
Framework	A set of software libraries that encompasses the application and executes it.
GPS	Global Positioning System - USA controlled satellite-based system to determine geoposition of the GPS radio signal receiver using time difference measurements.
GST	Global System of Telematics
HSDPA / HSUPA	High-Speed Downlink Packet Access / High-Speed Uplink Packet Access - Technologies developed on top of UMTS networks to improve user bandwidth.
HTTP	Hypertext Transfer Protocol

Term	Description
IPC	Inter-process Communication
J2EE	Java 2 Enterprise Edition
LAN	Local Area Network - A wired or a wireless network to interconnect devices situated nearby to each other.
LIN	Local Interconnect Network - One of the standardized ways of interconnecting vehicular electronic systems.
LTE	Long Term Evolution - 4G goal for the UMTS networks.
OASIS	Organization for the Advancement of Structured Information Standards
OBD	On-Board Diagnostics - An integrated system for detecting and logging faults in a car.
OSGi	OSGi Alliance is a global forum that standardized a Java-based service platform.
OWL	Web Ontology Language
Picocell	Cellular base station that covers a small area, analogous to Wi-Fi access points.
Platform	A runtime environment, or a set of APIs to build software products upon.
RDF	Resource Description Framework
RIA	Rich Internet Application - Application with a modern user interface that uses browser as the runtime environment.

Term	Description
REST	Representational State Transfer - A resource-oriented implementation style for Web Services.
RPC	Remote Procedure Call
RSS	RDF Site Summary - A method of providing XML based feeds of blog entries or news headlines over the Internet.
RTE	Runtime Environment
SOAP	A standard protocol for Web Services.
SVG	Scalable Vector Graphics - W3C standard for representing structured graphics.
Software agent	An independent application functioning on behalf of the user. Software agents are based on Semantic Web technology. See also: User agent.
Tekes	Finnish Funding Agency for Technology and Innovation
Third party services	Mainly Internet based services maintained by unaffiliated providers.
Ubiquitous Computing	A future vision, where information technology and computers are hidden in everyday objects and provide rich, context-sensitive and interactive services without desktop computers.
UDDI	Universal Description, Discovery and Integration - A directory of Web Services.
UI	User Interface

Term	Description
UMTS	Universal Mobile Telecommunications System - 3G long range telecommunications network technology based on GSM.
UPnP	Universal Plug and Play - Technology to interconnect devices in a local area network in an interoperable way.
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
USB	Universal Serial Bus - A plug-and-play bus to interconnect devices over a wired serial interface.
User agent	A client application or device mediating the interactions between the user and the web service, usually a web browser. See also: Software agent.
V2V	Vehicle-to-vehicle - Dynamic ad-hoc communication system between moving cars.
VII	Vehicle Infrastructure Integration - Initiative to connect vehicles electronically to roadside infrastructure so that overall road safety and efficiency can be improved.
Web Services	SOAP-based method of providing services over the HTTP protocol.
Wi-Fi	A wireless technology for local area connectivity between small devices and computers.

Term	Description
WiMAX	Worldwide Interoperability for Microwave Access - long range network technology for wireless devices.
WS-*	A family of extensions for Web Services.
WS-CDL	Web Services Choreography Description Language
WSDL	Web Services Description Language
WSDL-S	Web Services Semantics
XML	Extensible Markup Language - A general and interoperable serialization format for structured data.

1. INTRODUCTION

Carbook System is designed to bring Internet-enabled services into cars in a comprehensive and integrated way. Carbook System is associated to both the European Union and Tekes supported efforts towards ubiquitous computing [41], ambient networking [2] and intelligent transportation systems [30].

Carbook System is a distributed service platform in a sense that it provides the Internet-enabled services the necessary environment in which to operate. This includes, for example, service discovery and binding. Carbook System is also a software service framework, because it seeks to wrap the services into a consistent bundle with standard ways of executing the services. In this thesis, Carbook System can be referred to as Carbook Framework, or Carbook Platform, depending on the context.

Carbook System contains a number of open source, standardized and freely available interfaces and technologies, reducing the need to reimplement low level details. Carbook System is actually more like a selected set of existing technologies organized into a complete, consistent structure, than something completely new, built from ground up. This reduces the need to adapt the existing Internet services to function within Carbook System.

The goal of Carbook System is to provide a comprehensive infrastructure for car related Internet services, and therefore the scope of the undertaking is massive. Not only Carbook System includes the end-to-end connectivity, service discovery and binding, but also a huge semantic domain specification effort to ensure all the services are integrated to each other. The size of the effort means that the work must be done in an iterative way, so that every small milestone produces an incremental improvement to Carbook System.

Carbook System and related technologies will be demonstrated in a series of proof-of-concept demonstrations. The demonstrations are integrated in an automotive software infrastructure laboratory, Carlab. This thesis work consists of mapping and evaluating the essential technologies, and the initial implementation of the Carlab research environment. The thesis contains descriptions of the core technologies as well as a closer inspection of the implementation details for Carbook System and services. This should form a solid foundation for future incremental implementation and research work. The long term goal of the Carbook research effort is to acquire strategic competence and intellectual property in relation to this important technology field. The Carlab environment enables the competences and technologies to be developed and demonstrated.

Chapter 2 has a description of the background of the Carbook research effort and outlines the scope of this thesis work. Chapter 3 features selected key technologies and identified future trends, that will define the form of the Internet-enabled car of the future. Chapter 4 continues with a closer look on Web Services technologies, with special focus on Semantic Web. Chapter 5 features the draft design and some implementation details for Carbook Directory Service, a service that collects together the various services in Carbook System. Chapter 6 envisions some central services that will be enabled by the platform, with future implementation suggestions. Chapter 7 describes the runtime environment that enables services to be deployed in different domains of the infrastructure. Chapter 8 describes the user interfaces of Carbook System with focus on technical aspects and implementation details. Chapter 9 summarizes the status of the implementation of Carbook in Carlab. Chapter 10 concludes this thesis with some suggested future research topics.

2. STARTING POINT AND THESIS CONTRIBUTION

This chapter describes the starting point and the scope of the thesis work as it evolved throughout the associated timespan. Research nature of the Carbook research effort caused it to be strongly self-guided with no strictly defined targets or milestones. These milestones were, and are, constantly redefined as the direction becomes clearer.

2.1 Purpose of Carbook

The Carbook research effort is an attempt to find an intersection between different future visions in automotive and Internet technologies, and to bring together separate external research program goals. The name, “Carbook”, is derived from Facebook, as Web 2.0 and social networking technologies were immediately identified as the key focus areas. The Carbook research effort was started in the first half of the year 2008 as a set of small demonstrations and mock-ups to communicate the goals of the Carbook effort. One of such mock-ups is shown in Figure 2.1. The main focus of this thesis is on technology evaluation, while the Carlab implementation, started in this thesis, will be a continuing implementation effort with incremental demonstrations along the way.

2.2 Carbook Contribution

This thesis work was started with an initial purpose of forming a concept of Carbook. Additionally it became relevant to map out and evaluate the necessary technologies for Carbook System, and to design actual implementations for some of the Carbook services. As the work progressed, more key concerns were identified in this thesis work, such as device/service interoperability and Web Services technologies.

The Carbook effort was started in this thesis work, to be incrementally implemented in geographically distributed virtual and physical laboratory environment, Carlab. Carlab is a newly created environment that aims to emulate different domains in the automotive Internet service infrastructure. Carlab will include mock-

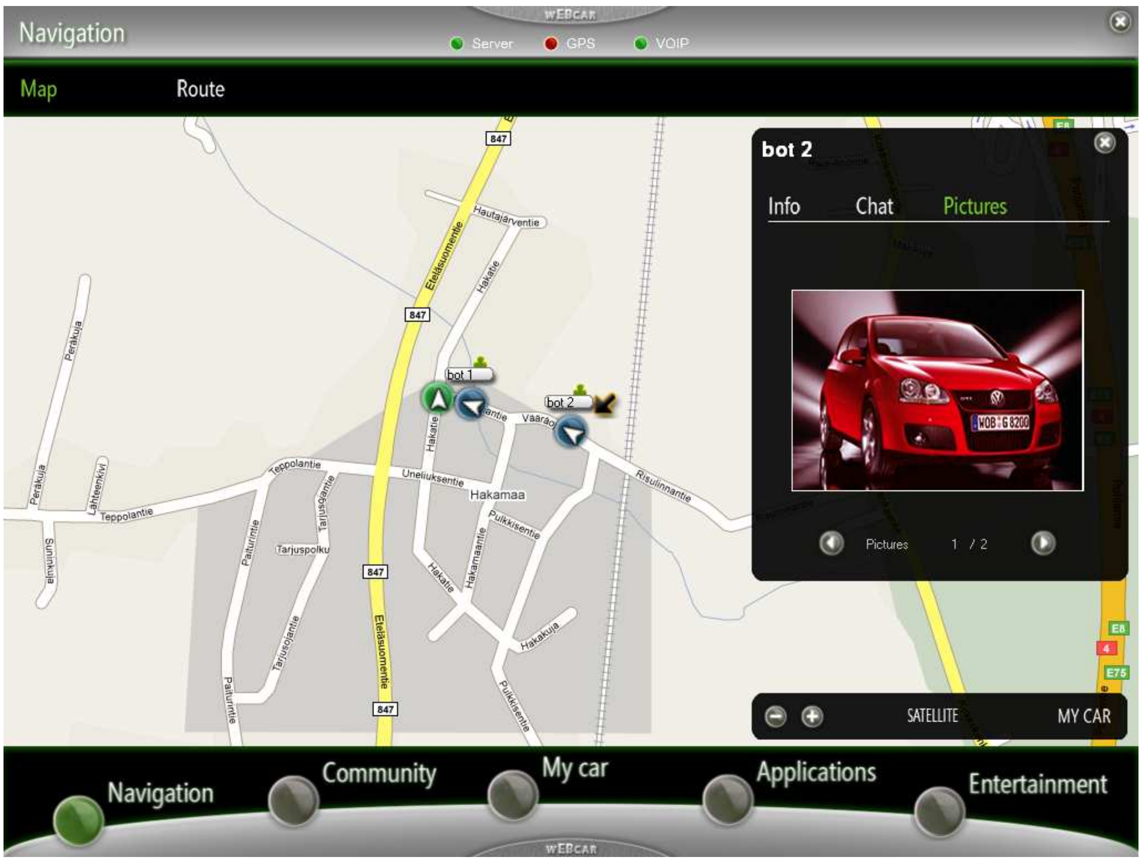


Figure 2.1: A mock-up for the Carbook concept

ups for home environment and car environment. The home environment will approximate a normal home of the end-user with modern home theatre equipment and ADSL Internet connectivity. The car environment will consist of a front part of a real car with heterogenous wireless Internet connectivity. These domains will be used in testing and demonstrating Carbook System. Design and partial implementation of the first phase of the Carlab environment was done in connection to this thesis. In practise, Carbook and Carlab are an inseparable, conjoined pair of concepts.

At the time of writing, Carlab is still under work, in relation to the actual layout of the laboratory space, and to the network topology to enable Internet services. The incremental implementation of Carbook System has been started and is expected to continue branching to many directions, as different implementations are being tried, tested and compared.

3. RECENT TRENDS AND KEY TECHNOLOGIES

Carbook System in whole is a web-enabled platform for mobile Internet services and independent software agents with support for both heterogenous terminal devices and special automotive user interfaces. Therefore, the key technologies identified in this research are in relation to modern web technologies and platforms, automobile communication buses, Semantic Web, independent software agents, device interconnectivity and interoperability, and mobile Internet.

This chapter outlines various key technologies and future trends. It is assumed that the reader is somewhat familiar with basic Internet technologies such as XML, HTTP and TCP/IP. Technologies closely associated with Web Services are described in Chapter 3.

3.1 Internal Automotive Communication Buses

Internal car electronic control units, ECUs, are interconnected by a car area communication bus. This bus is designed for small microcontrollers and simple electronic devices that do not have advanced capabilities. The car area communication bus is often based on either LIN (Local Interconnect Network, [9]), CAN (Controller Area Network, [8]) or FlexRay [3] standards.

LIN can provide data rates of up to 160kb per second, and is the cheapest and the lightest alternative of all three. CAN can handle data rates of up to 1 Mbit per second, being the standard most prevalent in the industry at the moment. CAN is also generally more expensive to implement than LIN bus. The new standard FlexRay can provide up to 10 Mbit per second date rates and is more expensive than CAN, and is not yet widely adopted. After the year 2008, it is required that all cars sold in the USA use ISO 15765-4 signaling [4], which is a variant of CAN.

In addition to the communication bus for the ECUs inside the car, other related buses have also been standardized, for example ISO 11992 [5], which defines a communication bus between towing and towed vehicles, and ISO 22902 [6], which defines the automotive multimedia interface.

Traditionally the automakers have been defensive in providing information about the internal car interfaces, such as on-board diagnostics (OBD). However, this is changing rapidly as legislation in the USA and in European Union will require the manufacturers to make these interfaces public. For example, the European Commission Directive 2002/80/EC [1] will require the automakers to provide interface information about the OBD systems upon request. Similar legislation to boost interoperability has been enacted in the USA.

Standard CAN buses do not have capacity to handle real-time video streams from cameras, and therefore this data is usually streamed over a different bus, such as Ethernet. The protocols over these buses have not yet been standardized in the automotive context, and it is not yet clear how tapping into the video feeds can be accomplished in practice.

Interfacing with the sensor and OBD data in the CAN bus is done through an OEM provided "smart gateway", which functions as a firewall between the safety-critical CAN bus and non-critical Internet-facing systems. In practice, it is expected that many automakers have deployed a Global System of Telematics (GST) compliant application server with a telematics API that can be used as platform for a small adapter application that publishes the relevant data to the applications running in the car runtime environment. An Internet-facing Web Service application can then perform aggregation, fusion, windowing, filtering and compression of data that can then be further published to a service provider.

3.2 Internet Protocol version 6

Current Internet is mainly based on Internet Protocol version 4, which was the first version of the Internet Protocol that was widely deployed. Global migration to Internet Protocol version 6, commonly shortened to IPv6, is underway due to scalability limitations present in the IPv4. For example, the pool of unallocated IPv4 addresses is forecasted to be completely exhausted between years 2010 and 2011. Always-on mobile devices with permanent IP addresses are a strong driving force in the migration. The larger address pool also enables more efficient routing.

The frame header structure for the IPv6 Protocol is depicted in Figure 3.1. There are many enhancements over IPv4 in IPv6, such as stateless address auto-configuration. This is a way of configuring network parameters to nomadic hosts with no state information maintained in the router. Stateless address auto-configuration is a clear enhancement over stateful Dynamic Host Configuration Protocol which centrally maintains states for each of the address leased to hosts in the network.

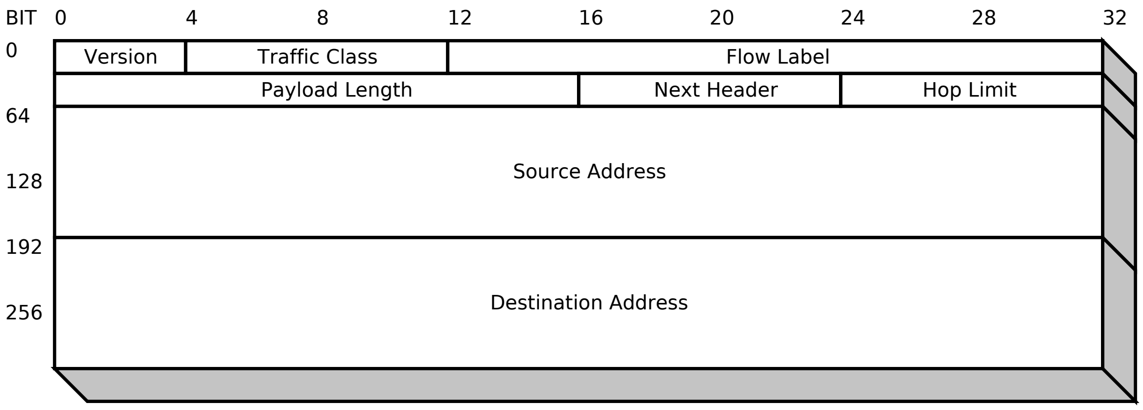


Figure 3.1: Frame header for Internet Protocol version 6 packets

Multicast is included in the base specification of IPv6, which guarantees the availability of the feature in IPv6 networks. Multicast is a way of transmitting identical data packets to a group of recipients so that the packets are sent only once over each packet link. The packets are only duplicated at the routers if recipients are reachable over different links. This greatly enhances the routing efficiency in IPv6 networks and makes possible a wide range of services depending on broadcast-type content such as high-definition Internet television.

In practice, IPv6 will enable full end-to-end connectivity, not only for cars, but also for the other Internet connected devices used in the mobile car context. However, cellular and other wireless networks do not yet have flat architectures necessary for seamless Internet connectivity or the necessary capacity to handle the envisioned amount of wireless Internet traffic. These problems are expected to be solved in near term by the operators, as the need for these features grows.

IPv6 has features related to Quality-of-Service and mobility that are especially relevant in mobile wireless applications. Wireless roaming and Mobile IP are described in more detail in the following sections.

3.3 Wireless Roaming

Roaming in wireless networks is one of the key aspects of mobility. Roaming means that the Internet connections and calls that the mobile device has created are not severed when the mobile device switches access points. Roaming enables the user to move within the wireless network freely.

Traditional solutions for enabling roaming functionality are generally implemented within the same access network. For example, GPRS/UMTS gives to the mobile device a stable IP address that does not change as the access point changes. This is implemented by network infrastructure that handles the complex routing task between the mobile device and the Serving GPRS Support Node router at the edge of the mobile operator network. This kind of roaming solution is know as "horizontal roaming", where the mobile device can roam within the area of the same network. This has also been implemented in IEEE 802.11, where the mobile device can roam between different access points without losing connections, as long as the access points belong to the same subnet.

Network agnostic roaming between different networks, for example between cellular network and WLAN network, is called "vertical roaming". Methods for vertical roaming have been standardized in Mobile Internet Protocol (IETF RFC 3775, [10]) for generic handover mechanisms and IEEE 802.21 for increased focus on Quality-of-Service aspects.

3.4 Mobile Internet Protocol

Mobile IP for IPv6 [45, 46] (IETF RFC 3775 [10]) is an extension built on top of IPv4 and IPv6 that allows the network interface to retain an externally stable IP address while the actual IP address used in routing changes as the mobile device changes access point. This is based on a home IP address where a home agent, functioning as a router, keeps track of the current IP address of the mobile device and routes the packets accordingly. The mobile device updates its new IP address to the home agent every time the IP address changes. This process is termed binding in the Mobile IP terminology, and the mobile device performs it by sending a binding update message to the home agent. The remote server where the mobile device is connecting to is called the "correspondence node".

Currently, router that is called an "Mobile IP router" is, as a rule, a Mobile IPv4 router.

In Mobile IPv4 there are two methods for routing. First, the packets going out from the mobile device to the correspondence node are routed directly, but the reply packets are routed through the home agent. This is known as "triangular routing" (Figure 3.2) and it is based on spoofing the sender address in the IP packets as the IP address of the home agent. Because of the nature of this spoofing, it sometimes causes the core network to drop the packets prematurely based on seemingly incorrect sender address. This is why most of the Mobile IP implementations offer the transparent routing option (Figure 3.3), where both the incoming and the outgoing traffic is routed through the home agent, which is non-optimal. In any case, Network Address Translation (NAT) like functionality is needed to maintain numerous

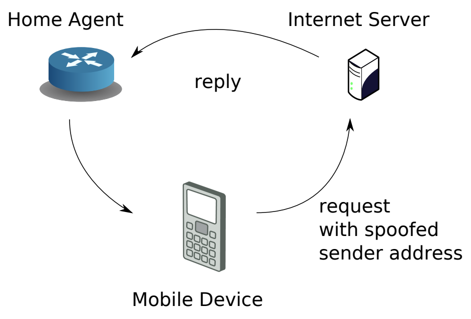


Figure 3.2: Triangular routing

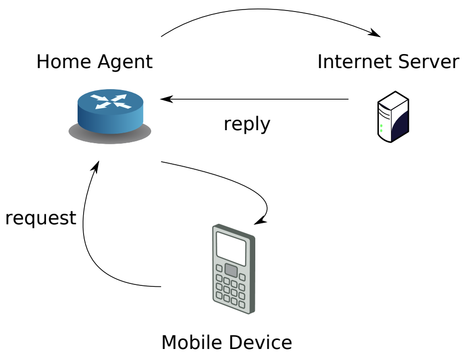


Figure 3.3: Transparent routing

mobile devices. This makes mobile servers and direct connections between mobile devices problematic in Mobile IPv4.

Mobile IPv6 has some advantages against Mobile IPv4 as Mobile IPv6 uses a special Mobility field in the packet headers to enable the mobile device to update its address directly to the correspondence nodes, as shown in Figure 3.4. The correspondence nodes then authenticate the new IP address of the mobile device by using the “Return Routability Procedure”. This means that the mobile device will prove to the correspondence node that both of the two IP addresses are routed to the mobile device. The authentication is required to prevent an attacker of spoofing the mobile IP address to gain access to the mobile session, or denial-of-service. This procedure of authenticating the nomadic IP address is detailed in the IETF RFC 3775 - Mobility Support in IPv6 [10]. It should be noted that bulk of the traffic will be routed directly between the mobile device and the correspondence node, not through the home router, as is the case of IPv4.

The most important difference between Mobile IPv4 and Mobile IPv6 is that Mobile IPv6 enables mobility for servers, and seamless connectivity between mobile devices.

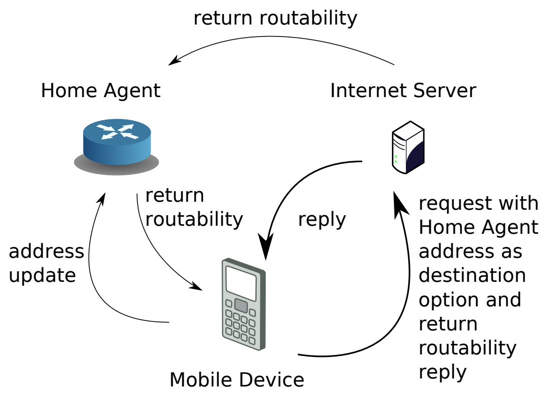


Figure 3.4: Internet Protocol version 6, optimized routing

Since Mobile IP is not a proper standard, nor is it widely deployed at the time of writing, it is expected that the implementations wary considerably and interoperability issues are expected. However, because of the nature of Mobile IPv6, it is possible to fall back to the process of routing all the packets through the home agent if the destination is otherwise unreachable. This makes the special roaming routing completely transparent to the core network and to the correspondence agents, while falling back to non-optimal routing case.

There exists a push towards flat architectures in future cellular networks and therefore lighter roaming technologies, such as Mobile IPv6, warrant closer examination. IP mobility support in 3GPP2 networks is already based on Mobile IPv4.

Internet connectivity in cars is visioned as being network agnostic in such a way that different wireless networks and technologies can be used as they are available, possible even simultaneously. Some of the wireless networks might be nomadic, which means that they have no integrated roaming support. This means that the IP address changes when the access point changes. The IP address also changes inevitably when one radio access technology is switched dynamically to another for example from GPRS/UMTS to WiMAX. Mobile IP technologies enable roaming while the actual IP address changes, thus providing completely network agnostic roaming.

Mobile IP has a Network Mobility extension [12] that enables whole networks to roam between access points. This functionality is a central aspect of the related IST Ambient Network project sponsored by European Commission within the Sixth Framework Programme. Network Mobility will be needed in car domain networks that have a single, changing Point-of-Attachment to the Internet. Additionally there is some work underway to enable fast handovers in Mobile IP in IETF RFC 4068 [11]. A free Linux implementation exists for fast handovers in Mobile IP [32]. Fast handovers in roaming are critical for real-time applications such as Voice-over-IP.

3.5 IEEE 802.21

Mobile IP is a rather simple technology with almost no special network infrastructure required. It can also work transparently so that externally no handovers are visible. However, Mobile IP does not handle real-time traffic and Quality-of-Service gracefully. For example, the handover initiation is not specified in the Mobile IP RFC. These concerns have caused a new standard, IEEE 802.21 [7], to be created which enables QoS-based soft "vertical handovers" with QoS and handover initiation processes between heterogenous networks. Unlike the Mobile IP technology, IEE 802.21 requires special support from the network infrastructure. IEEE 802.21 compliant infrastructure is projected to be deployed between years 2009 and 2010 [43]. The scale of future deployment is not yet known, and Mobile IP provides a convenient fallback sacrificing soft handovers.

3.6 Universal Plug-and-Play and Digital Living Network Alliance

Universal Plug-and-Play (UPnP, [27]) is an interoperability standard for different multimedia devices. The standard constitutes a couple of different device profile standards that determine how the devices will be discovered and controlled in a local area network. The discovery domain of the UPnP devices is usually a local area wireless network. One of the important new features of the UPnP-protocol is the device's capability to offer a web user interface for the purpose of controlling the device.

At the moment, most of the UPnP-compliant devices are targeted to trusted local area network connectivity, and the lack of security and authentication layer in the UPnP stack makes it inconvenient to offer services outside the local area network. It is evident that the UPnP will evolve in near-term towards Internet-enablement utilizing IPv6 and Web Services extensions. Devices Profile for Web Services (DPWS, [26]) is a successor of UPnP that is completely aligned with Web Services technology, and therefore DPWS is fully Internet-enabled technology.

Digital Living Network Alliance (DLNA, [28]) is a global collaborative group of companies. Digital Living Network Alliance has developed a set of interoperability guidelines and a conformance certification process on top of UPnP-standards. There are many DLNA-compliant devices in the market, for example Nokia N95 mobile multimedia computer.

3.7 Scalable Vector Graphics

Scalable Vector Graphics (SVG, [20]) is an XML based web standard for describing vector, or object, graphics as serialized, human-readable strings. SVG 1.1 is a W3C recommendation. The difference between raster graphics and vector based graphics is that raster images are represented by an array of pixel values, while vector images are represented by a tree structure of objects. This lack of concept of a pixel causes vector graphics to be infinitely zoomable, and therefore particularly

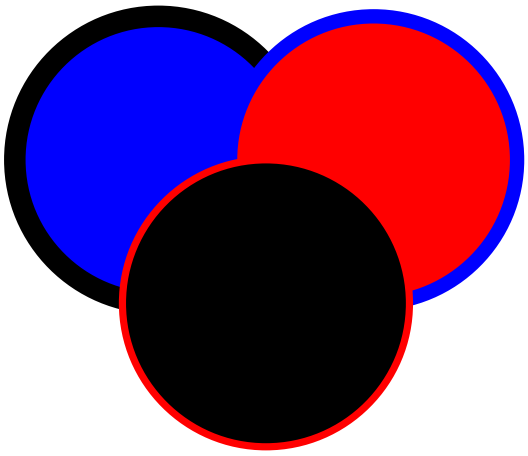


Figure 3.5: Scalable Vector Graphics sample image

Listing 3.1: SVG serialization for Figure 3.5

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
 "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="100%" height="100%" version="1.1"
  xmlns="http://www.w3.org/2000/svg">

  <circle cx="35" cy="35" r="20" stroke="black"
   stroke-width="3" fill="blue" />
  <circle cx="65" cy="35" r="20" stroke="blue"
   stroke-width="2" fill="red" />
  <circle cx="50" cy="55" r="20" stroke="red"
   stroke-width="1" fill="black" />

</svg>
```

useful in certain applications. It is also possible to associate some semantics, or meaning, to structures in the image, which is not possible with raster images.

The vector images are shown by rendering them on the target medium pixel-by-pixel, as raster images. Figure 3.5 gives an example of a SVG image rendered to target medium, which is either paper or computer display, depending on whether this thesis is an electronic, or printed on paper. The SVG serialization of the image is given in Listing 3.1.

SVG graphics are especially useful, as they can be naturally composed together, and therefore they make a great data visualization tool. For example, it is trivial to draw line based data on a map to visualize a region of the world. The existing SVG graphics can also be dynamically modified, for example by changing line width or color, to visualize selections and other user interface features.

In the context of Carbook System, SVG is expected to be heavily used on mash-ups of cartographic and geographic data, for example in drawing GPS tracks and positions of interest on top of the base layer map. Rich capabilities of SVG enable visually impressive graphical user interfaces maintaining strict platform and display medium independence.

4. WEB SERVICES

Web Services is a way of deploying Internet-enabled services utilizing XML-based technologies to enable easy integration and wide interoperability between different services. Web Services technology is widely used in enterprise systems, because it enables relatively painless integration with well-defined interfaces for the Internet-enabled services. This chapter will outline the most essential Web Services technologies and their use in the Carbook framework.

4.1 Servlet Container and Web Services Platform

Figure 4.1 illustrates a typical configuration of a Web Services platform. On top of the operating system, a web server and a database are deployed. The web server handles and relays incoming HTTP-requests and usually handles some basic authentication and logging functionality. The web server alone is able to serve static or simple dynamic web pages. The database is used to store the data of the service, and it is usually deployed to a separate physical server, or to a cloud of servers.

The Java 2 Enterprise Edition (J2EE) servlet container handles service component life cycles and maintenance functions. The J2EE servlet container registers itself to the web server to listen the HTTP requests. The servlets deployed in a servlet container will often need to use database connections, which are usually

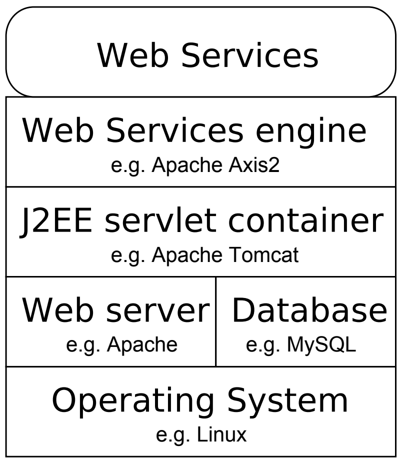


Figure 4.1: A typical platform for Web Services

Web Services Extensions	WS-Addressing, WS-Policy WS-Security, WS-*
Web Services	SOAP, WSDL, XML Schema
Web Server	HTTP
Network	TCP + IPv4/IPv6

Figure 4.2: Protocol stack for Web Services

maintained by the servlets themselves. The servlet container itself does not generally need a database to function. Apache Tomcat includes a HTTP-server of its own, so the Apache web server is not strictly necessary for the example configuration. However, it is possible to use a separate web server for performance reasons as depicted in Figure 4.1.

On top of the servlet container, it is possible to deploy a Web Services engine as a servlet. The Web Services engine handles the life cycles and management of different Web Services and provides them Web Services functionality.

The general protocol stack is outlined in Figure 4.2. Internet connectivity is handled by TCP/IP, but in some cases such as in Devices Profile for Web Services, UDP/IP can also be used. The web server accepts HTTP connections from clients. These HTTP connections contain SOAP Web Services Remote Method Invocation calls and other Web Services messages. Web Services have a number of extension protocols that enable specific functionality in addition to the basic Web Services capabilities.

4.2 Web Services Extensions

Web Services standard is extensible by nature, and many extensions have been created. Some of these extensions are WS-Security, WS-Reliability, WS-Reliable-Messaging, WS-Addressing, WS-Transaction and Web Service Semantics. These are often referred to as WS-* specifications.

While the pure WSDL is supposedly a serialization-agnostic description language, the WS-* extensions are as a rule strongly bound to SOAP serialization.

WSDL describes the syntax for web services, while leaving the service semantics outside the specification. Consequently, two different web services might have exactly similar WSDL interfaces, but they might be for two completely different purposes. For example, if a service takes a value of type xsd:string as a parameter returning a value of type xsd:float, it does not give any hints for independent software agents about the purpose of the service. One such service might be for getting stock quotes, and another for checking prices of products. Web Service Semantics (WSDL-S, [15]) extension, WSDL-S, enables semi-automatic processes, application integration and independent software agents by providing the semantics for the WSDL-defined web service interfaces. WSDL-S is partly based on the work done in OWL-S standardization, and in fact these two standards are overlapping. Both of the standards provide a method of communicating the semantics of a web service to independent software agents, but WSDL-S is a standardized extension to WSDL, and so it is better integrated into Web Services than OWL-S standard.

4.3 Web Services Description Language

Web Services Description Language [13], WSDL, is a core component of Web Services technology. WSDL defines an XML-based language for defining the syntactic interfaces for Web Services.

WSDL interface specifications are often automatically generated from the program source code, or the other way around, the program source code is generated from the WSDL interface specification. Automatically creating WSDL interfaces from the source code has a downside of causing unstable interfaces that can change with a change in the source code or when a tool to create the interface is upgraded or changed. This is categorically against the idea of the interfaces being independent of the internal implementations of the composing components and this makes automatic generation of the interfaces inadvisable. However, there are reasons for choosing bottom-up approach, for example, to integrate some existing systems into the enterprise architecture, to remove the need for the developers to learn and use multiple different languages for a single application, or to speed up the development cycle in projects that do not need extremely stable interfaces between the services. This is often the case in the small in-house web application projects.

4.4 Web Services Choreography Description Language

Web Services Choreography Description Language [14], WS-CDL, is an XML-based language for describing interactions, or choreography, between different business parties when web services are divided between multiple participants. Examples of such divided web services include credit card transactions, travel agents and Business-to-Business e-processes. WS-CDL offers an unambiguous way to communicate the technical requirements for the divided services, enables automatic validation of business logic and reduces dependencies and constraints between the business partners and between the web service platforms used.

Carbook System aggregates together a large number of web services of variable complexity from different third party providers. These services are often directly coupled with the business processes of the companies offering them. Sometimes the business process spans multiple companies working together to offer the service, and thus the service is actually divided between independent parties. One example of this kind of division could be a service for ordering car accessories by mail. The complex business process might consist of an intermediary service that gives the user a list of compatible accessories filtered by user preferences. This service might then confirm the user identity from a specialized service, and relays the order with relevant information to a third company which provides the actual accessories. The third company will need to charge the user by negotiating payment with a financial institution, and then mail the order using the services of a logistic company. The company providing accessories might later pay a monthly commission for the customers acquired through the intermediary company. When the complete business process spanning multiple companies is automated, it will benefit from a formal description of the interface agreements between the companies.

4.5 SOAP and Representational State Transfer

SOAP was once an acronym for Simple Object Access Protocol, but this was dropped with version 1.2 for being misleading. According to Sanjiva Weerawarana, a speaker in Google Tech Talks: “SOAP never had anything to do with object access, it was not a protocol, it was a message format and it certainly isn’t simple” [52] .

SOAP is created for decentralized and distributed environment for the purpose of exchanging information in an interoperable manner. SOAP is based on XML and has two main parts:

- header, that is the metadata associated with the body, and
- body, that contains the actual message.

SOAP is a core part of Web Services technology.

Representational State Transfer, REST, is a convention of structuring web services as a collection of stateless resources with common interfaces. This structure was introduced in 2000 in the doctoral dissertation of Roy Fielding [42]. Web services structured so that they follow Fielding’s REST principles, are commonly called RESTful. RESTful web service resources have a Uniform Resource Identifier and they are generally used through HTTP even if this is not strictly necessary. The resources implement a subset of four different actions commonly mapped to HTTP methods with the same names: POST, GET, PUT and DELETE. These methods are related to CRUD-operations (Create, Read, Update and Delete) commonly used in databases. It should be noted that only the HTTP get is used in HTML links and HTML forms use both HTTP get and HTTP post methods. The PUT and DELETE methods [50] are not currently included in the HTML standard, while there is a clear need of adding them. At the moment, the PUT and DELETE methods are triggered using scripts or applets embedded in HTML documents such as JavaScript in AJAX.

A large portion of the traditional web services are "accidentally RESTful" in a sense that they conform weakly to the REST principles without implementing them fully. This is because of the fact that the REST is derived from the way HTTP and the web work today, while SOAP is an additional layer built on top of the HTTP.

SOAP and REST are partly complementary and overlapping Web Services technologies for interaction between a user agent (a browser) and a web service. It does not cause a considerable overhead to support both the REST and SOAP interaction models. WSDL, WS-CDL and WSDL-S can be used seamlessly with both the SOAP and RESTful services. It is also possible to structure SOAP Web Services to be completely RESTful, that is, a service can be structured as a resource. The main difference between SOAP and REST is that SOAP is meant for Service Oriented Architectures and REST is meant for Resource Oriented Architectures. Commonly accepted guidelines as to when it is better to use the one or the other have not yet emerged.

4.6 Extensible Messaging and Presence Protocol

Extensible Messaging and Presence Protocol (XMPP/Jabber, [21, 22, 23]) is an Instant Messaging (IM) protocol that is freely extensible through XML. This protocol has been born for use in Jabber IM network, but has found rich applications elsewhere as well. Nowadays, the protocol is competing with HTTP as a transport layer for service oriented architectures.

XMPP has gathered growing interest as a Web Service transport [24, 53, 54]. One of the key players that has invested heavily in XMPP is Google. Google uses XMPP as a base for it's IM and VoIP-services [35].

The primary use for XMPP is Instant Messaging, and there are many public Jabber servers in the Internet for this purpose. They generally provide transporting service to other IM networks such as ICQ, so that the user presence is delegated to multiple IM networks simultaneously. This means that the user is available for messaging in multiple IM networks at the same time. Many companies have their own, private XMPP/Jabber servers to facilitate internal instant messaging.

HTTP provides a traditional, client-server foundation for web services. Client-server architecture does not scale well in P2P (Peer-to-Peer) style environments, because the HTTP lacks one critical ability: The ability to push data to clients. This disability has led to polling style interaction, that can rapidly saturate networks. HTTP-based services are usually based on opening new connections for every interaction, because the connections are often targeted to different servers. This often causes unnecessary overhead.

XMPP allows persistent connections between P2P partners, and this makes almost real-time messaging possible. The connections are always on, and the messaging between clients is asynchronous, which removes the need for polling.

The communications between the service and clients goes through the XMPP server, so that XMPP is not really a pure-blooded P2P technology. However, XMPP is often used to communicate references to sidechannel datastreams (for example, HTTP hyperlinks), which can be used to transfer data straight from service to clients without the XMPP server in the middle. This makes the routing between nodes more optimal for bulk data, allowing direct P2P-connections. However, firewall considerations should be taken into account, as clients might block different types of sidechannel data. Sometimes, it is possible to send large chunks of data through the XMPP server, when all the more efficient methods are blocked.

4.7 Semantic Web

Semantic Web has been the main goal of web standardization work done at W3C organization lead by Tim Berners-Lee. Semantic Web is an evolutive step forward from the current “syntactic web” towards decentralized information. The vision of Semantic Web is to make the web browseable not only by humans, but also by independent software agents. Semantic Web will enable advanced discovery and aggregation of global services and resources. Semantic Web will be mainly built on RDF and Web Services.

Semantic Web in full will not realize in many years yet, but the technologies developed are already extremely useful in integrating services of a limited domain together. For example, the Dublin Core ontology is already widely used in document formats and web pages. Carbook System will be structured so that it will both speed up the adoption of Semantic Web technologies, and gain advantage from the aspect of automatic integration and aggregation of services. It would seem that the next step in several fields, such as knowledge management, media interoperability and ad-hoc networking would be exploiting the Semantic Web technologies to achieve higher levels of abstraction, interoperability and seamless integration.

The domain specific ontologies that enable Semantic Web functionality in Carbook System consists of an open set of several independent domains, for example:

- road topology,
- road regulations,
- points-of-interest and routes-of-interest for communities,
- tourism and information,
- entertainment and events,
- media, and
- weather.

Some of these domains are already sufficiently described by existing semantic models, but it is expected that new ontologies are needed. The continuous work should concentrate on searching and reusing existing semantic models and augmenting them as needed. Creating new ontologies should be the final option, when sufficient models do not exist yet.

4.8 Independent Software Agents

Independent software agent has many definitions, in this thesis it will be defined as a software application that works on behalf of the user towards a clearly defined goal by utilizing the services and resources it encounters while crawling the web. These software agents may be simple applications running on any static platform, for example on a mobile device or on a server of a service provider. The agents may be shared between multiple different users, particularly in the case where the goals of the different users are not in conflict. The independent software application also may or may not migrate and multiply between different runtime platforms to accomplish goals.

Software agents have been used mainly in web search engines, but are increasingly gaining new opportunities as the web is evolving more semantic traits. In Carbook System, the independent software agents are seen as a tool to reduce the direct dependence on user input in the Carbook services. This frees the user to concentrate for example on driving while the independent software agents are working on behalf of the user.

In principle, the independent software agents are seen as method of raising the abstraction level of the human-machine interaction. Instead of selecting the keywords "weather in Tampere" and entering them to Google, and then browsing through the search results and linked web pages, perhaps the user could only state an intention to show weather information, which the agent could then find in different formats applicable to the car domain context. For example, the weather could then be shown as an overlay on top of navigational map, or spoken by synthesized speech.

The real power of independent software agents comes from truly independent action; For example the agent might constantly search different map overlays from Carbook Service Directory and different other sources. These overlays can then be aggregated to a browseable form available to the user. Similar agents could browse Internet radio stations and RSS news feeds to find interesting content for the user.

4.9 Resource Description Framework and Web Ontology Language

Resource Description Framework (RDF, [16]) and Web Ontology Language (OWL, [17, 18, 19]) are languages for describing web ontologies, endorsed by World Wide Web Consortium. Actually, OWL is a general knowledge representation language often serialized in RDF/XML syntax, while the RDF was originally created to be a metadata description language, now used to represent general knowledge also. RDF represents knowledge in triples of subject, predicate and object. OWL ontologies consist of axioms that enable making complex inferences about the individuals (classes) and their properties. In a way, OWL provides an inference engine that can be used with RDF described resources.

The basic principle behind Semantic Web is that the information is structured as resources, as opposed to documents in the syntactic web, that have associated semantics included which makes all the information connected to each other in meaningful ways. Resources are not necessarily reachable over the Internet, but they can still be referred to from Semantic Web. This enables automated reasoning through web query languages so that it is possible for the user to make complex searches into the global knowledge pool and for the independent software agents to perform advanced tasks on behalf of the user.

The common view towards Semantic Web technology has been somewhat pessimistic, and many big hurdles have been identified along the way before the final vision of Semantic Web will realize. However, it has been noted that while the final vision of Semantic Web will still take years to materialize, the standardization work done already is very useful for example in service oriented architectures making the integration of complex systems with multiple independent participants manageable.

4.10 Universal Description, Discovery and Integration

Universal Description, Discovery and Integration (UDDI, [25]) is an OASIS standard with a number of stable implementations that aims to enable locating web services by robust queries against rich metadata.

A UDDI server can be thought as a phone book that enables searching electronic Web Services descriptions and associated metadata, such as information about the service provider. UDDI registration consists of three parts:

- white pages, for address and contact information of the provider,
- yellow pages, for industrial classification according to standard taxonomies, and
- green pages, for technical interfacing and binding information about the services provided by the business (WSDL).

It is possible to use UDDI directories only as electronic phone book replacements with no Web Services bindings, but in this thesis, it is necessary to concentrate specifically to electronic, Internet-enabled services.

UDDI will be in a central role in Carbook Service Directory, where the Web Services can be naturally indexed, searched and advertised. In practice, a UDDI directory service implementation, such as jUDDI, can be deployed as a servlet on top of a servlet container, for example Apache Tomcat. In the Carbook context, it will be necessary to create a custom taxonomy of automotive domain electronic services instead of using existing Yellow pages taxonomies, such as the North American Industry Classification System (NAICS), the Standard Industrial Classification (SIC), or the United Nations Standard Products and Services Code (UNSPCS).

5. CARBOOK DIRECTORY SERVICE

The service-oriented Carbook architecture needs to be scalable, decentralized and robust. The services are aggregated together from various sources, one of which is Carbook Directory Service. This directory service in turn collects together services that have certain trustworthiness, and are safe for the user to interact with. It also serves as an open portal into Carbook System.

Because of Carbook Directory Service, these services will form a consistent bundle that is easy to navigate and use even in the constrained in-car environment using heterogenous terminal devices.

5.1 Context-Aware Services

Carbook System shown in Figure 5.1 consists of services that are made available over the Internet. Carbook Directory Service enables services to be integrated into a consistent, dynamically structured bundle.

To facilitate the context-aware and inter-domain services, the car domain service environment is associated with the corresponding home service environment and the services provided by the automaker in the Internet. For example, all the music and movies shared in the home domain are implicitly available in the car through home domain media services. The car fault diagnostics, sensors and logs are available through the car domain services. The automaker and third party service providers might offer some extra services such as route finding, traffic alerts and RSS feeds that are available over the Internet. All these services are accessible by the user over the Internet and collected together by Carbook Directory Service.

Carbook Directory Service publishes the lists of services and maintains associations between different domains. In principle, one user has a Carbook identity and associations to number of domains, for example to user's home and car. A number of users can associate to same domains, in other words, the domains of the users can overlap. Associations between domains are user specific. In practice, the association has two of mechanisms.

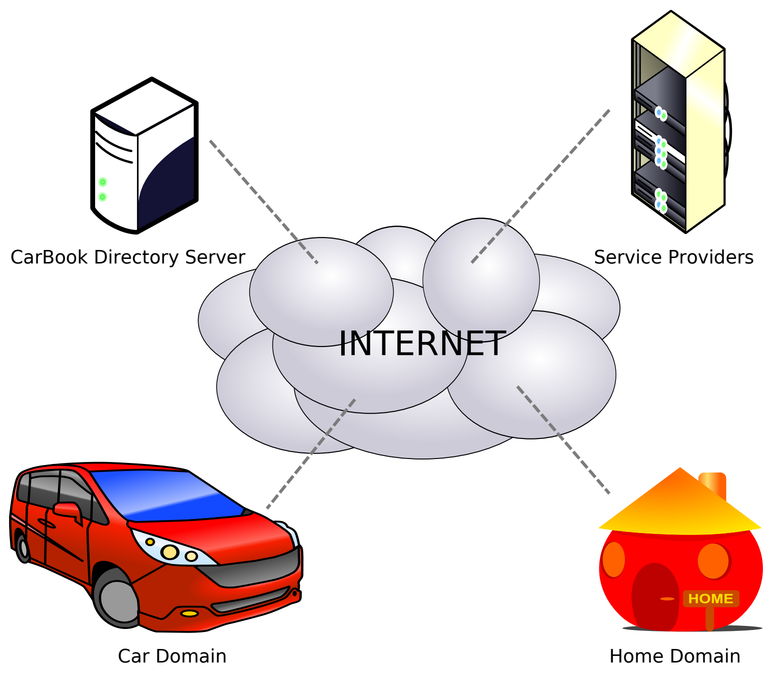


Figure 5.1: Carbook System

Firstly, the implicit association comes through the act of accessing the service through different domains. The Internet-based service maintains its state even if it is accessed using different methods. This means that, for example, the files downloaded to a media service are available to all the domains for playing back. Implicit association is based on user identity, which associates the different sessions together in standard Internet services.

Secondly, the service gets extended user identity information if it needs it, from Carbook Directory Service, which includes the list of associated domains with access descriptors. This access descriptor is actually a list of IP-addresses with relevant metadata, such as the type of domain in question, to enable the service to contact the services provided by the different user domains. This could be used for example to make an Internet call or send a chat message to a certain user so that the call or the message is routed to all the associated domains at once. This also enables the user to connect to and to use the services provided by, for example, the car domain, while interfacing Carbook System from his home. Access to the extended user information is, of course, controlled.

The linkage between the user identity information, and the associated car domain, and the home domain, also separates the domains from each other so that the services can become context-aware, and that different set of services can be offered to different domains (Figure 5.2). In essence, Carbook Directory Service will offer

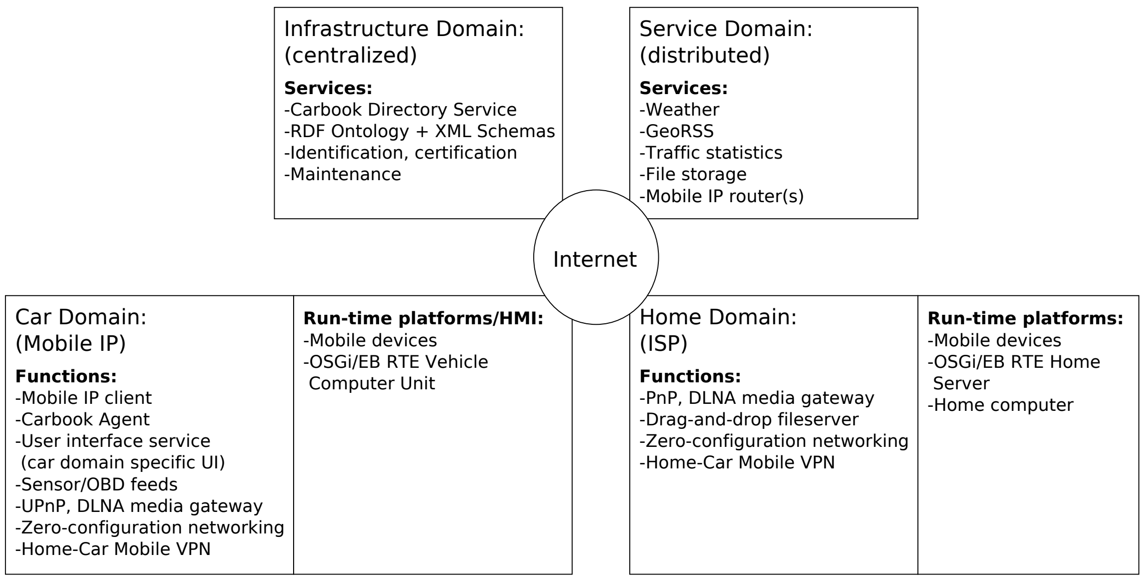


Figure 5.2: Carbook domains

different views to different users and to different contexts. This necessitates that the domain of the user of the service, whether the user is a human being or a software agent, is known by the service. In practice, this information can be given when connecting to the service. Additionally, this requires a taxonomy of different domains, albeit a simple one in this case, and easily implemented using ontologies. Implicit context-sensitivity can be implemented using local data storage and internal services that are visible only locally, inside the domain.

The domains are also separated by encapsulating the internal services so that they are externally invisible. This means that domains have an external interface for the services made available to the Internet and to other domains. Access control is implemented using standard authentication and security measures.

5.2 Semantic Service Aggregation

It is expected that there will be a great number of services available to the end user from a multitude of different service providers. These services can build upon each other, and provide meta services on top of existing services, for example, by providing a centrally maintained list of streaming media services. Administrating these manually with traditional methods simply will not be feasible or scalable in the long term. This necessitates the use of Semantic Web technologies where possible to aggregate and fuse the services together forming a consistent whole.

Carbook Directory Service is based on RDF, WSDL, Web Services and XML. Carbook Directory Service stores and makes available a set of WSDL service descriptors, through UDDI, that can be dynamically browsed by the user or by independent software agents. The WSDL service descriptors are added into the system by different parties, for example the end-user may add services to be privately available in the home, in the car and over the Internet. Third-party service providers can add their own applications into the system so that they are globally browseable by all the users.

Every service in Carbook Directory Service is represented by a WSDL service descriptor and associated metadata, stored in a UDDI system. Every service itself is an ordinary Web Service that can be hosted anywhere on the Internet, including the car domain and the home domain.

Semantic Carbook RDF ontology is used to describe the service semantics so that the service descriptors can be effectively searched, aggregated and used by crawlers. This is necessary to facilitate independent software agents that remove some of the burden of interaction from the user who needs to concentrate in driving. This also enables more seamless integration of heterogenous services.

5.3 Trust and Security

Carbook Directory Service is centrally administered by Carbook Directory Server. This Carbook Directory Server maintains a trusted certificate database with access permissions. For example, a certified service provider is able to add globally visible services and applications to Carbook Directory Service, while a single user can add applications and services for private use only. The architecture is open in a sense that applications and services are freely available on the Internet, and the user does not need to adhere to the Carbook provider maintained list of trusted services.

The service provider signs the service descriptions with the certificate granted to it. The Internet Web Service itself does not need to be in any relation to the service description provider, and any Web Service is in principle directly integratable into Carbook System without any modifications just by providing the semantic Web Service descriptor to the UDDI directory. Also, while a certification process is necessarily needed for the service providers to receive a revocable but retransmittable certificate, services and applications can be added to the directory freely by any certified party. For example the case of user installing a new service into Carbook System might be simply downloading a service descriptor from the non-certified service provider and uploading this into Carbook Directory Service. This makes the service bookmarked for this user through self-certification.

5.4 Using the Directory Service

There are multiple use cases for Carbook Directory Service, most essential being the service browsing case. The availability of other use cases, such as maintaining the list of services, domain associations and user identities, is based on roles in the system. Access to different functions and views are limited and filtered according to user role and identity.

Users and independent software agents are able to browse and discover these services and compose a user specific list of bookmarked services that form the basic application tree for the user. Carbook Directory Server maintains this user specific list of bookmarked services so that the view to these services is persistent and available to the user in different domains.

Every user has always the minimal set of services that are built-in to the system. On top of these services, there might be some default services pre-bookmarked for the user by the automaker, or by a third-party service provider.

The built-in services are needed to manage the service bookmarks and applications in the system and to provide an initial user experience with a possibility of customizing the bookmarks and services heavily to suit the tastes and preferences of individual users.

The directory service will provide a dynamic tree view of the available and bookmarked services to the user, where the services can be directly interacted with. In practice, the services can provide a rich Internet application, or plain HTTP user interface that can be accessed through the Carbook Directory Service user interface. Some services might have a known Web Service interface type, such as streaming media services, that can use predefined user interface. This means that these basic service types can be more seamlessly integrated into the desktop or operating system user interface. The subject of user interfaces in Carbook System is discussed further in the Chapter 7.

6. CARBOOK SERVICES

Carbook includes different kinds of services that are discovered through Carbook Directory Service. Modelling the system as a collection of services makes the system extremely scalable. New services can be easily created on top of existing services by combining them in new ways. All services are made available in the Internet as Web Services to facilitate platform independence and maximum flexibility. The Representational State Transfer (REST) model and SOAP are supported and used side-by-side for the Carbook services and the services are published by Carbook Directory Server with Web Services Description Language (WSDL) and semantic metadata. This chapter outlines preliminary visions of the services provided and drafts some of the implementation details.

Implementor of the described services should take into account the relevant security considerations, such as user identity protection, security of communication channels, and protection against the misuse of systems. However, these considerations are left outside the scope of this thesis.

6.1 Service model

The services function either independently in the background, launching events or logging data, or are query-response oriented passive services. Active services can be turned on and off by the user or by a software agent representing the user through the service interface. When functioning independently, the services can be thought as agents that are functioning on behalf of the user. Extensive use of independent, user authorized agents is needed to free the user from any and all unnecessary interaction, where this interaction may pose a safety risk while driving.

Services in Carbook System are divided into:

- user created services, and
- third party services.

User created services are home domain or car domain based services that are either deployed and activated by the user or are integrated into the system. These services provide a Web Services interface that is accessible over the Internet. These services can be limited to be used by the user so that the services normally accessible in home domain only become accessible though the associated car domains also. This is the case with the home domain media service that makes available the UPnP stored media files to the associated car domains. The first-party services can also be used by third party service providers for example for the purpose of gathering sensor data from the cars. The initiative for the user to provide such data might be receiving a value-added service from the service provider in return.

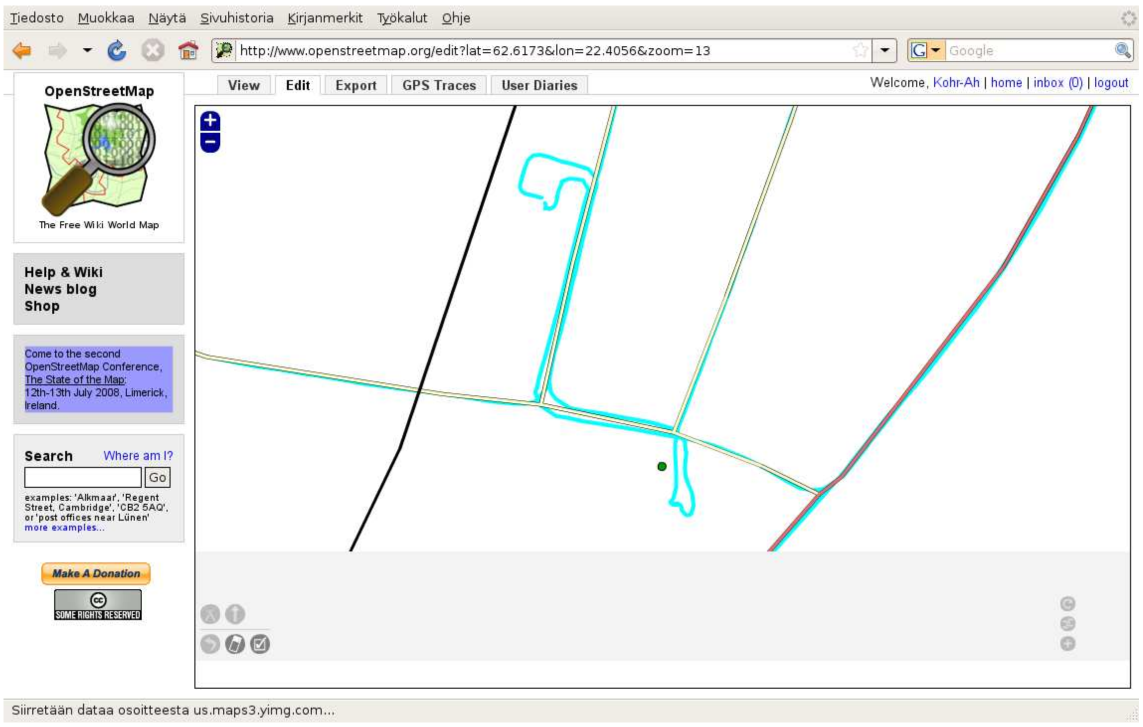
6.2 User Interface Service

Services that are intended to be used by people can provide Rich Internet Application (RIA) User Interfaces over HTTP. This makes the services available for users through commonplace heterogenous Internet terminals.

For a richer set of interactivity that is normally not possible with RIA, the service can also bind to a domain specific User Interface Provider Services. There should be User Interface Provider Services inside the car to provide services access to specialized user interface devices such as IC card readers, webcams and microphones for speech input. These are especially necessary in contrived Car Domain, where it cannot be expected that the user can access the services using the normal keyboard-/mouse input model. In-car user interfaces must be designed to require minimal amount of concentration to use so that they do not pose unnecessary safety risks to the user. The actual implementations of these interfaces will be heavily influenced by the automakers, and are expected to vary considerably between different makes and models of cars. The user interfaces of Carbook System are discussed further in the Chapter 7.

6.3 Car as a Sensor

Modern cars can have tens of sensors and this number is bound to go sharply up in near future. These sensors include fault sensors, temperature sensors, fuel level sensor, cameras, speedometer, odometer, GPS receiver and other sensors. Car measures its environment and its internal state actively, generating a constant data stream. This data could be useful for a variety of purposes, such as fault diagnostics, travel diaries, weather and traffic forecasting, and reality harvesting. Figure 6.1 shows an example of a stored GPS track being imposed on a map from OpenStreetMap [34]. This GPS track can be used to create new map content. Car as a Sensor service is


Tiedosto Muokkaa Näytä Sivuhistoria Kirjanmerkit Työkalut Ohje
http://www.openstreetmap.org/edit?lat=62.6173&lon=22.4056&zoom=13
Google
View Edit Export GPS Traces User Diaries
Welcome, Kohr-Ah | home | inbox (0) | logout
OpenStreetMap
The Free Wiki World Map
Help & Wiki
News blog
Shop
Search Where am I?
Go
more examples...
Make A Donation
SOME RIGHTS RESERVED

Come to the second OpenStreetMap Conference, The State of the Map: 12th-13th July 2008, Limerick, Ireland.

examples: 'Alkmaar', 'Regent Street, Cambridge', 'CB2 5AQ', or 'post offices near Lünen'

Siirretään dataa osoitteesta us.maps3.yimg.com...

Figure 6.1: GPS track of a vehicle imposed on a map from OpenStreetMap

a car domain service that makes all this data available on request. The Car as a Sensor service can also trigger different alerts based on preconfigured conditions.

Logs of data, stored fault codes or data streams can be queried from the Internet by any parties that are authorized to do so. For example, some geolocation services might need a constant access to user's GPS coordinates to deduct the road the user is driving on and use this information to filter notifications the user has subscribed. The Car as a Sensor service is also made available wirelessly in-car. For example, it is possible to watch any integrated camera feed using a wireless mobile terminal by a car passenger.

Media type services such as camera feeds are made available in-car using UPnP and DLNA technology so that they are freely available to any device supporting these protocols. Tunneling these sensor feeds to the user's home domain is automatic and requires no special interfaces.

Offering Car as a Sensor services without restrictions to the Internet and to third-party service providers may pose a security and privacy risk, and so a separate Internet facing interface is required. The Car as a Sensor service is visible to the Internet as a web service that accepts connections from the Internet with access restrictions. For example, the car manufacturer might have retransmittable remote access to the car sensors over the Internet, so remote fault diagnostics become pos-

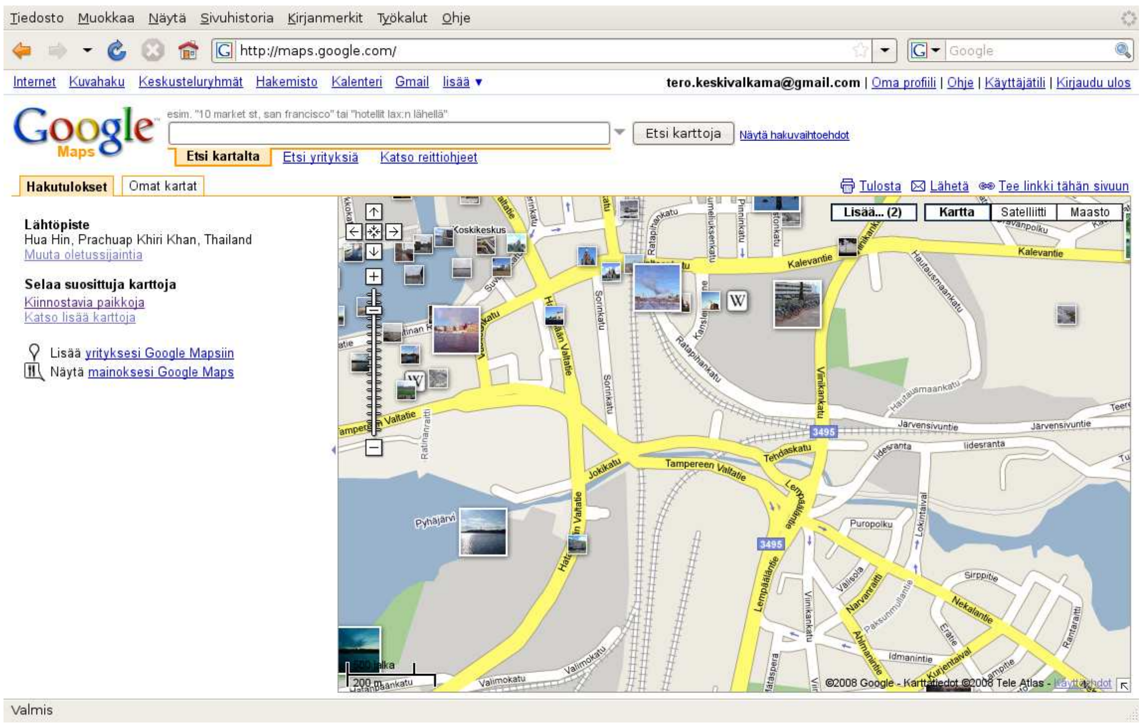

Tiedosto Muokkaa Näytä Sivuhistoria Kirjanmerkit Työkalut Ohje
http://maps.google.com/
Internet Kuvahaku Keskusteluryhmät Hakemisto Kalenteri Gmail lisää ▾
tero.keskivalkama@gmail.com | Oma profiili | Ohje | Käyttäjätili | Kirjaudu ulos
Google Maps
esim. "10 market st, san francisco" tai "hotellit lax:n lähellä"
Etsi karttoja
Näytä hakuvaihtoehdot
Etsi kartalta
Etsi yrityksiä
Katso reittiohjeet
Hakutulokset
Omat kartat
Tulosta
Lähetä
Tee linkki tähän sivuun
Lähtöpiste
Hua Hin, Prachuap Khiri Khan, Thailand
Muuta oletussijaintia
Selaa suosittuja karttoja
Kiinnostavia paikkoja
Katso lisää karttoja
Lisää yrityksesi Google Mapsiin
Näytä mainoksesi Google Maps
Lisää... (2)
Kartta
Satelliitti
Maasto
Koskikeskus
Kalevantie
Tampereen Valtatie
Järvensivuntie
Iidesranta
Pyhäjärvi
Tehdaskatu
Puropolku
Nekalantie
Lempääläntie
Sorinkatu
Ratapihankatu
200 m
©2008 Google - Karttatiedot ©2008 Tele Atlas - Käyttöehdot
Valmis

Figure 6.2: Google Maps map of Tampere with overlaid GeoRSS

sible. It should be possible for the user to grant explicit permissions to third parties to read these feeds and logs.

6.4 GeoRSS

The current de facto GeoRSS technology is the Keyhole Markup Language (KML, [31]) developed by Google Inc. This makes it possible to bind the RSS (RDF Site Summary) entries to certain locations so that they are shown only when they are visible on the map region shown. This functionality is demonstrated in Google Maps in Figure 6.2. The KML is mainly interesting to the user in laying a selected set of GeoRSS feeds on top of a navigation map. Carbook System provides a way for the GeoRSS feeds to be discovered and subscribed over the Internet by leveraging Semantic Web technologies.

6.5 Mediaserver

Carbook System handles media sharing by connecting heterogenous media sources through a defined set of services. Primarily, two different types of sharing are considered:

- media streaming between domains, and

- file synchronization between domains.

A drag-and-drop service for moving files such as documents and music files between domains is planned as a separate service. This service would receive the files from the user through a web interface in a drag-and-drop manner into different directories representing the different domains. When the associated domain becomes online the next time, the contents of the directories will be synchronized into the local storage space in the said domain. This service will enable the user to move documents and media files between different domains. This service can be naturally implemented using Web-based Distributed Authoring and Versioning (WebDAV, [29]).

Additionally, UPnP and DLNA compliant home media systems offer a plug-and-play media streaming service. UPnP packets can be selectively tunneled between the different domains to make the user media stored in a home media center available in the car while driving. UPnP compliant devices used in the car domain can discover the media streaming services locally even though they are actually remote. Remote tunneling may cause latency and bandwidth issues that would not be evident in a real local area network, and this may necessitate filtering out some of the UPnP streams that would require either high bandwidth or low latency. This could also be solved by pre-buffering, or downloading the streamed music or video locally to the car, possibly filling up the otherwise silent delays with locally existing content.

6.6 Fault Codes / Diagnostics

Practically every modern car has some fault diagnostics integrated with fault codes that can be downloaded from the car by a mechanic at a licensed car repair shop. In addition to normal OBD fault codes, other information can also be relevant to diagnosing problems, for example:

- car / owner information, service book, mileage,
- route logs,
- fuel consumption logs, and
- user created diary of servicing and fault related events.

This diagnostic information is made available to the Internet using a dedicated in-car service that listens queries or sends triggered alarm messages to pre-specified service providers. This information can be then read by the user or by the car mechanic authorized by the user.

6.7 Social Technology / Messaging

This service handles the user profile and messaging details, such as nicknames, icons and other data that should be carried over from one social service or application to another. The thriving of the Short Message Service, SMS, in cellular networks has demonstrated how important it is to not be limited into voice communications. Today, Internet and web enabled technologies, such as email and instant messaging are rapidly gaining ground over SMS messaging and phone calls. These services are rapidly integrating into Voice-over-IP services and social networks. In a way, it could be said that the user identity is converging in these different communication services into one unique identity that can be contacted in a variety of ways. In any case, the speech communication, in synchronous and asynchronous forms, will be of special importance in the limited car environment, where the use of a keyboard might not be possible.

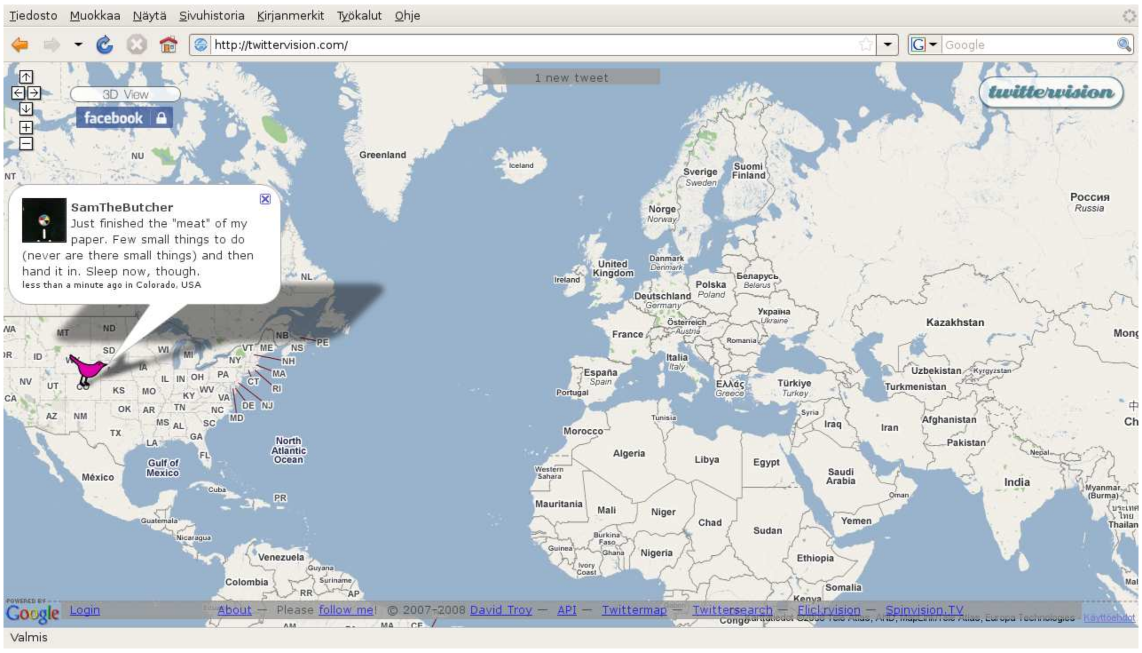
The vision of Carbook is bringing the Facebook into the car so that the communication method is more or less transparent to the user. Existing convergence technologies are used and brought into the car to achieve this goal. The communication between people is divided into separate functions:

- browsing and finding the people,
- checking the status and availability of the person to contact, and
- establishing synchronic or asynchronic communication link between the users.

Browsing and finding people is being rethought in social networks, whereas mobile technologies still rely on personal phone books stored in the memory of mobile phones and centralized directories of names and phone numbers. Still, there should be no more need for showing the user the phone numbers, or email addresses of the people that there is need to show IP addresses of the services in the Web.

The status of the user is converging in the different kinds of instant messaging services with client programs delegating the status of the user into separate instant messaging networks. Car environment brings some new statuses for the user such as “driving”, or “traveling”, that have special meaning to the communication networks. These should be implemented using separate ontologies to achieve wide interoperability without the burden of traditional standardization processes.

User status communication has been lately enhanced to encompass messaging from individual to community, such as in the Twitter service [38]. The Twitter service is based on short messages, “tweets”, that communicate what the user is doing right now. This type of service is called “micro-blogging”, and the two main


Tiedosto Muokkaa Näytä Sivuhistoria Kirjanmerkit Työkalut Ohje
http://twittervision.com/
Google
1 new tweet
twittervision
3D View
facebook
Login
About — Please follow me! © 2007-2008 David Troy — API — Twittermap — Twittersearch — Flickrvision — Spinvision.TV
Valmis

SamTheButcher
Just finished the "meat" of my paper. Few small things to do (never are there small things) and then hand it in. Sleep now, though.
less than a minute ago in Colorado, USA

Figure 6.3: Twittervision showing real-time "tweets" on the world map

providers of micro-blogging services are Twitter and Jaiku [39]. In principle, micro-blogging is an evolution step from predefined, taxonomied states of status, to free form status messages. This small piece of information can be further associated with other data, such as geoposition, to create new services. One example of such secondary service is TwitterVision, a service that visualizes real-time "tweets" on the world map (Figure 6.3).

Asynchronic messages, such as voice email, are of special relevance in the car environment, where the user might not have access to a keyboard, or possibility to react to the communication immediately. It is easy to imagine that voice email will become a very important communication method in cars in the future.

Social networks get a new dimension with Carbook System, when it is possible to give access to the real-time data streams in your car to your friends, and upload the car collected data into different services to be used by virtual communities. It is visioned that hyper-social behavior could and should be brought and encouraged into the road environment, and enabling direct social interaction between drivers could decrease, or even eradicate, different kinds of negative behavior such as driving under influence and road rage.

7. APPLICATION RUNTIME ENVIRONMENT

In Carbook System, there are two different runtime environments for Web Services applications:

- home domain runtime environment, and
- car domain runtime environment.

The services integrated into Carbook System can also run on user's mobile devices, on home computers and on various server platforms in the Internet, but these are not considered to be parts of Carbook System. This is because these platforms are heterogenous, non-standard and non-interesting from the perspective of the system. However, the services can be added to Carbook System by separate devices connected to the Internet, whether they are inside the car, in the user's home, or provided over the Internet by a third party service provider.

This chapter will outline the principles and methods to be used when the runtime environments, and applications enabled by them, are implemented.

7.1 Car Domain Runtime Environment

Car Domain Runtime Environment (Car Domain RTE), is a platform for Semantic Web Services applications to be run in the car domain. The home server in the user's home is also a Web Services platform with a different operating context. It is visioned to be highly similar to the Car Domain RTE, without the APIs specific to the car domain. Service deployment on the home server is left outside of the scope of this thesis, because this functionality is not critical in respect to the complete system at the moment.

Deploying the Web Service applications in the car reduces latencies and makes the services available in offline mode also. The application running on the Car Domain RTE can also get special access to certain Application Programming Interfaces, APIs, that are not feasible to use remotely over the Internet mainly because of security, latency and bandwidth issues. Also, migrating full, or thin client Web Services applications to the Car Domain RTE is made possible to make reduction

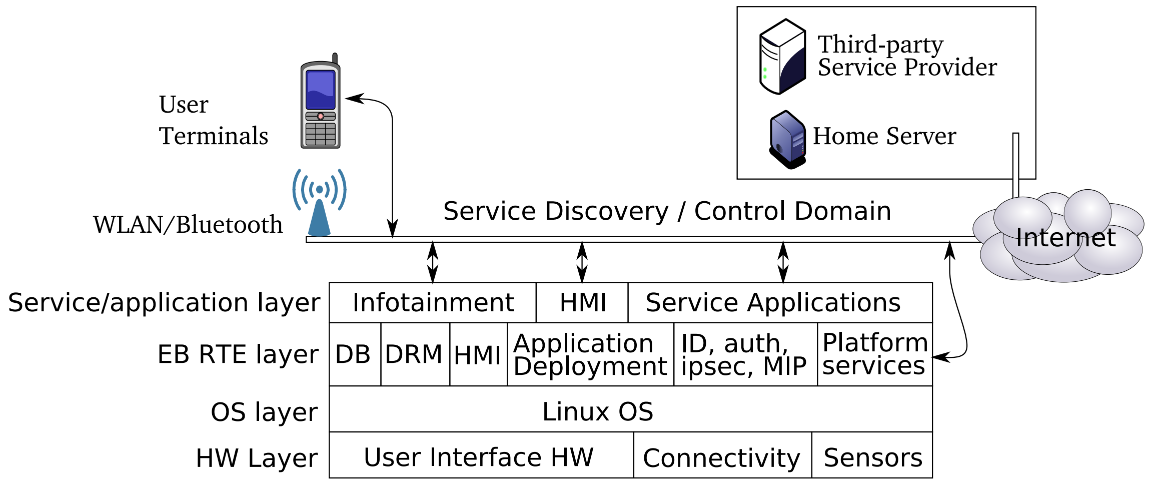


Figure 7.1: Car Domain Runtime Environment

of the bandwidth consumption on some bandwidth intensive services possible. The draft structure of the Car Domain Runtime Environment is depicted in Figure 7.1.

Car Domain Runtime Environment is a service-oriented platform based on Java technology [37] and OSGi platform [47]. Car Domain RTE makes use of OSGi methods for IPC, packaging and deploying service components. This makes it possible to decrease the application dependence on the underlying operating system, makes available a large amount of components already made to OSGi-platform and leverages the mass of people already familiar with this type of environment. Java and OSGi also have a good support for Web Services and XML which are the core technologies in this architecture.

Car Domain Runtime Environment works together with legacy infotainment systems and communicates with the car Electronic Control Units, ECUs, through the security media gateway as shown in Figure 7.2. The legacy systems can use Carbook services such as UPnP zero network configuration capabilities, but this is not strictly necessary. Legacy systems may also provide some additional services and Carbook can function as an adapter for these legacy services, so that they will be integrated seamlessly into the Carbook architecture.

7.2 Existing Car Domain Platforms

There are many different platforms for developing applications for the car computer. OSGi Alliance [33] has produced specifications for service application deployment built on top of Java Archive Technology [44]. The OSGi platform [47] consists primarily of a framework for application life cycle management and a service registry, applications running on top of Java Runtime Environment. The application life cycle

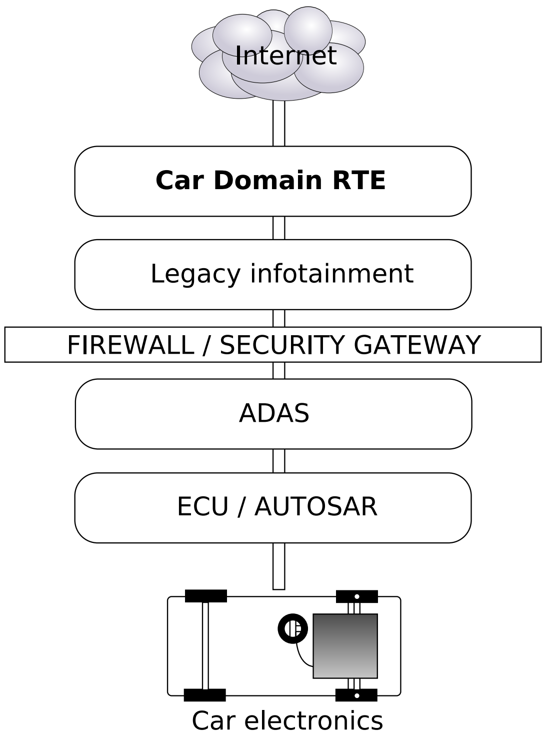


Figure 7.2: Relation to legacy infotainment systems and ECU

management needs to be taken into consideration when service applications from multiple different service providers need to be separately installed, started, stopped and uninstalled without affecting the rest of the system. Applications are considered as separate components, "bundles", with separate life cycles and dependencies on each other. The service registry [49] is a concept for service discovery and binding used inside one virtual machine, and it seeks to replace the Java Listener Pattern [48], which is widely used, but does not scale well in embedded environments [49].

This platform is designed for smart home appliances [51], automotive telematics, Internet gateways and consumer electronics and offers a flexible way of distributing and updating applications installed on Carbook System while not compromising security without specially built in-house solutions. OSGi platform is widely used in the industry.

Global System for Telematics [36], GST, is an EU-funded Integrated Project that, among other deliverables, provides specifications for a GST Open Systems platform for controlling and reading automotive telematics. GST Open Systems compliant platform provides access to car sensors and to Internet services. The GST project has ended in February 2007.

7.3 Component Technologies

Providing a runtime environment for a heterogenous set of components with different life cycles necessitates some centralized control for keeping the system in whole responsive and operating correctly. From the perspective of the platform, this means sandboxing the service applications and access controls to administrative functions such as deploying new applications, removing existing ones and granting new permissions to applications. In principle, the application should be allowed to access the public services provided by other applications and by the platform, but not to interfere with the normal operation of the system. From the perspective of the application provider, this means that the applications must conform to general guidelines and interface specifications. If special permissions are required for the application to operate, these are acquired from a certification authority by cryptographically signing the application as conformant to requirements to the permissions in question. Similar application certification processes are in use in other operating systems as well [40] and they are also being incorporated in multipurpose consumer operating systems.

Car Domain RTE uses the platform services provided by the OSGi platform for application deployment and life cycle management, so that the application components can be dynamically installed and removed without affecting the other applications or rebooting the system. Additionally, a style guide should be provided later to facilitate interoperability between applications from different third party providers.

8. USER INTERFACE

The users of Carbook System are either owners and passengers of cars, or service providers. In some cases these roles will inevitably overlap, since normal users are allowed to integrate their own services into Carbook Distributed Service Directory. Carbook System is used through web user interfaces enabled by Rich Internet Applications.

Because of the nature of the Carbook services, different kinds of user interfaces are required. These types of user interfaces are reiterated in this chapter.

8.1 Rich User Interfaces

Basically the terminal equipment and the service provider can choose any web user interface technology, such as Microsoft Silverlight, Ajax, Flash, Java, native client application or basic HTML. Some of these technologies are compatible with different terminal devices than others and they vary in richness of user interfaces also.

In a service oriented architecture we will not need to make services and terminal devices uniform in such way that any service would be usable through any terminal. This means that Carbook System in whole will be seen as a small collection of simple and specialized services for simple specialized devices and on the other hand, a large collection of generic services for terminals that support these. Every device takes a complete list of services available and filters out those it is not capable of interfacing with. On the other hand this means that the service provider does not need to support a broad range of devices if that would be too costly, but only choose to support certain classes of devices.

There needs to be some kind of method for choosing the best user interface available for the user's terminal device. Of course, if a native client application is available for the terminal, this should be used, as it is tailored for this specific terminal type. Native applications can also bypass some technical limitations that might be problematic through pure WWW interfaces. For example, it is not trivial to pass microphone input through to a vanilla WWW service without a special client-side application. Other special equipment such as joysticks, cameras, IC card and fingerprint readers might necessitate the use of native client application also. While the native client applications have their undeniable advantages, they also have a lot of disadvantages. It is often impossible for a service provider to support more than a handful different client applications for different terminal types. This problem will become more pronounced because of the sheer number of different kinds of mobile devices compared to two or three main flavors of operating systems on home computer field. It is also harder to update the applications that are installed to a client device compared to a pure web service that can be upgraded on the server side.

Depending on the type of the service, it is possible to expand the user base by providing more generic user interfaces without compromising the user experience by leveraging RIA technologies. As said earlier, these will inevitably have less control on the user device than a native client application. However, full control is rarely needed, because most of the services can be easily implemented using standard web user interface components without low level access to the hardware.

8.2 User Interface Service

When a richer set of interaction is needed in addition to keyboard and mouse input paradigm, it will be necessary to interface with some built-in user interface devices in the car. This can be used for speech-to-text dictation, VoIP calls with image and IC card payments. Some devices might allow for multiple access such as microphones, but others, such as full screen video on an integrated video screen, might require some locking.

From a service point of view, interfacing to these User Interface Services is similar to interfacing car sensor services. Every service needs an explicit authorization to use these external user interface devices, and this authorization might be pre-signed into the service bundle by a trusted authority, or explicitly granted by the end-user. UPnP stream services are exempt from this authorization process, because their nature is inherently public.

8.3 Universal Plug-and-Play

UPnP services can be for example audio and video streams. These do not need Carbook System to operate, but Carbook System can provide a hospitable environment for these devices to operate in by converting UPnP and Bluetooth services dynamically into Carbook Services by publishing them in Carbook Directory Service and tunneling them from one location to another, and the other way around, advertising Carbook Services in the UPnP and Bluetooth domain as applicable.

With some services, it is appropriate to make them first UPnP and DLNA compliant, and then advertise them in Carbook Directory Service with necessary application components handling the tunneling and access control between locations. This approach has several important qualities; The local services can be used locally without any signaling between Carbook Directory Server and the place-of-use, and it also integrates all the existing UPnP and Bluetooth devices seamlessly into Carbook System.

Tunnelling of the UPnP services between different domains is initiated by the user. User can use the local user interface to browse the locally available UPnP services and mark them up to be tunneled to other domains if appropriate.

8.4 Software Agent Interface

The independent nature of services imply that the user might authorize an agent to function in her behalf towards some well defined purpose. This means that practically all the user interfaces that are accessible to the user, should also be accessible to independent software agents. This means that the service interfaces need to be machine accessible over the Internet and that the services need to have machine readable semantic service descriptors associated to them. The purpose of the software agent in this sense is to gather and aggregate services into more useful combinations and abstractions. Often the service aggregator needs the authorization of the user to do this, and this makes it a software agent functioning on behalf of the user.

The metadata needed for this automatic interaction with services is included in Carbook Directory Service, and is the same information that is used in enabling service interoperability.

Examples of this software agent interaction would be automatic searching and indexing of Internet radio stations using the preferences and access rights of the user, and finding the events and interesting placemarks in the locality of the user, perhaps leveraging the friend networks of the user and the communities the user belongs to.

9. CARLAB IMPLEMENTATION

Carlab is a physical laboratory space that is the environment for incremental implementation of the Carbook System. It will also be a central location for internal and customer demonstrations of the capabilities of the Carbook System. Carlab is built from scratch, starting from work space layout, by deploying new servers and services to incrementally demonstrate Carbook functionality. The first milestone for the Carlab will be in Fall 2008, and is expected to achieve the basic Carbook functionality.

This chapter outlines the network topology of the Carlab and summarizes some of the services that have been already implemented, or that are currently being implemented.

9.1 Mobile IP

Mobile IP routing is a necessary precondition for any meaningful mobile Internet services. For demonstration purposes, Mobile IPv4 home agent is being deployed in the Carlab, with roadmap plans to upgrade it to full scale Mobile IPv6 support.

The Mobile IP home agent is first demonstrated with GPS Trace Extraction application installed to a group of mobile phones. The GPS Trace Extraction application is paired with a Mobile IP client to send binding update messages to the Mobile IP home agent, when the IP address changes. Demonstrations of GPS Trace Extraction with Mobile IP are expected to start in Fall 2008.

Mobile IP home agent resides in the public portion of the Carlab network topology, and is visible to the Internet.

9.2 OpenStreetMap Integration

Location can be visualized in many ways, the simplest of which is a simple coordinate. However, a simple coordinate is not very descriptive for a human observer, because the location described is not directly evident. For example, software bugs in the system might not be directly recognizable, when the location is given as a simple coordinate.

To facilitate the demonstration of location dependent services, there needs to be a way to visualize locations. This is normally done by drawing location indicators on top of a map background. In the Internet, there are numerous maps services available, some of them more open to free commercial use than others. For demonstration purposes, Carlab project is using OpenStreetMap services to provide the needed map backgrounds.

The backgrounds are published as map tiles, rendered to pictures from vector data. These tiles are queried by the client from the map tile provider, or repository, when certain sections of the map are needed. At the client side, the map tiles are collected and rendered to the correct positions. Location information and other augmenting data can then be visualized on top of the generated background by the client.

Often, the map visualization engine is built as a web-enabled application, so that visualizing different data on top of the background map can be done, for example, with Javascript and Scalable Vector Graphics.

In practice, the client needs to have data sources to visualize on top of the map. These data sources could be, for example, GPS traces, current GPS locations, GeoRSS feeds and other location dependent data. These data sources are visible to the user as layers that can be selected and deselected freely, showing and hiding the associated data as needed. The data sources have different types, that need explicit support from the client. At first, a simple GeoRSS support is planned to be demonstrated along with custom GPS trace information collected from 3G mobile phones. Later, data source discovery and general support for multiple types of data sources will be implemented.

The location visualization is functionality, that is targeted to the user's perspective of the Carbook System. In practice, this means that the visualization engine is available from the Car domain and from the Home domain of the Carbook System. For this purpose, the Carlab includes the Home domain and the Car domain mock-ups along with necessary network infrastructure.

Demonstration of the OpenStreetMap are expected to commence in Fall 2008 with several other milestone demonstrations.

9.3 GPS Trace Extraction from 3G Mobile Phones

Position dependent services rely on accurate, real-time positioning data from the clients. To be able to demonstrate these services, some GPS feeds must be available to the Carbook System. For this purpose, a client software was developed to 3G

Access network	Wired / Wireless	Remarks
Intranet	wired	corporate intranet
Visitor (wired)	wired	primarily for visitors
Visitor (wireless)	wireless	primarily for visitors
ADSL	wired	consumer ADSL line
Static, public IP	wired	For Carlab

Table 9.1: Summary of the access networks available in Carlab

Access network	Firewall	Authentication
Intranet	HTTP proxy, firewall, NAT	HTTP Proxy
Visitor (wired)	NAT	none
Visitor (wireless)	NAT	WWW authentication form
ADSL	none	none
Static, public IP	none	none

Table 9.2: Access networks available in Carlab, security characteristics

mobile phones using Python and XML, to stream measured GPS positions to the Carbook Position Data Store service.

The Carbook Position Data Store is an implementation for publishing the collected coordinate data for the secondary services. Web Services interface and Carbook Directory Service integration for the Carbook Position Data Store are future milestones for these services.

The GPS Trace Extraction capability is first demonstrated with integration to public OpenStreetMap service, by overlaying the real-time data and collected statistics on top of the map background. Demonstrations for the GPS Trace Extraction are expected to start in Fall 2008.

9.4 Network Topology

There are a number of different networks accessible in Carlab physical area. These are shown in Table 9.1 and in Table 9.2. This was the starting point when the Carlab network implementation was started.

It was decided that the Carlab network is to be separated from the visitor and intranet networks for security reasons. The implemented network topology is shown in Figure 9.1, and described in Table 9.3, and in Table 9.4.

The implemented Carlab network consists of Carlab LAN subnet and two local IPv4 subnets, Carlab VPN and Carlab WLAN. Switches and some of the less important routers are left out of Figure 9.1. The Carlab also has a number of static global IPv4 addresses, that are publicly accessible from the Internet. The public IP

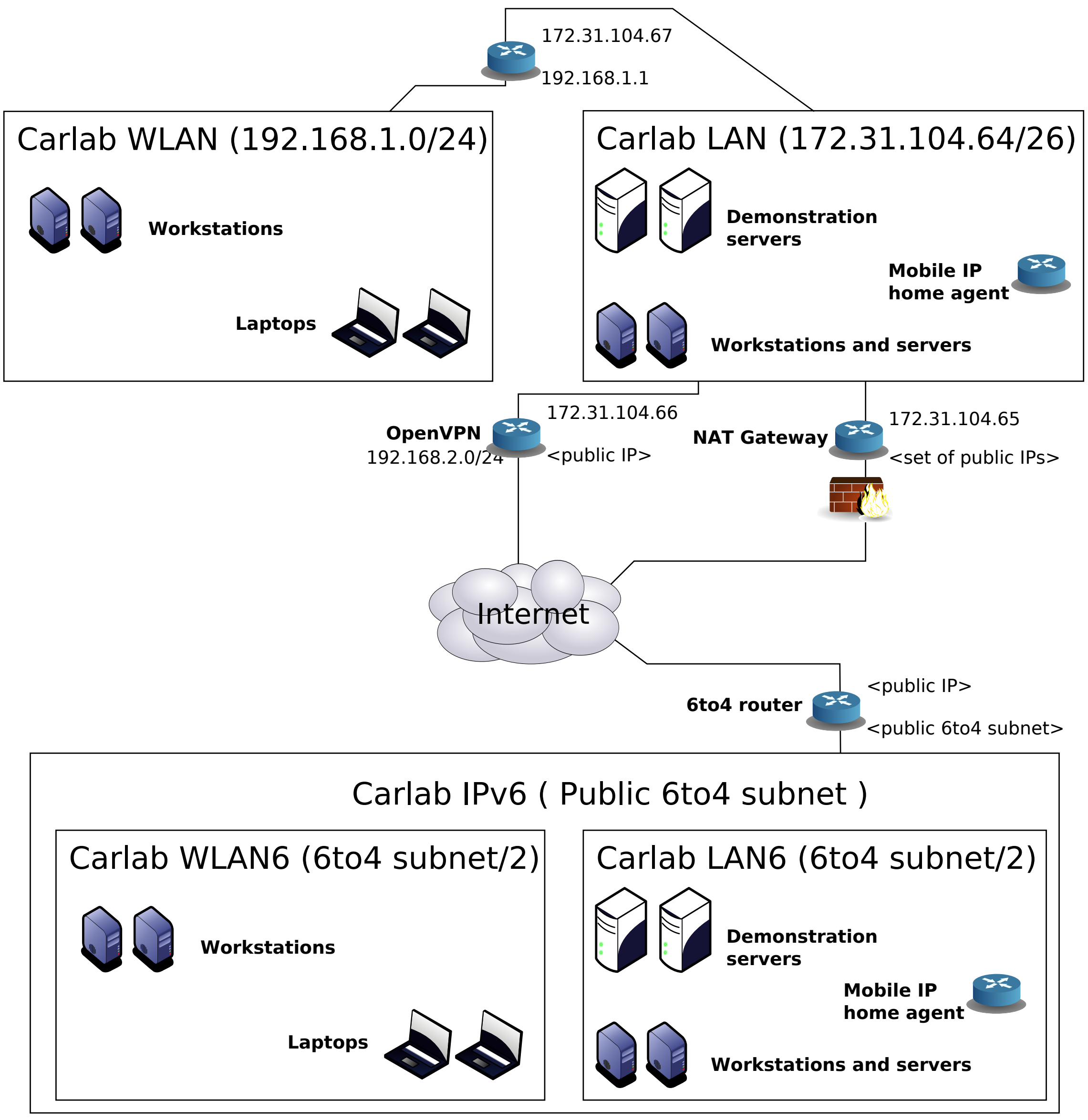


Figure 9.1: Network topology in Carlab

addresses are left out of this thesis for security reasons. The IPv6 is one of the key technologies to be evaluated in Carlab, and so twin networks, Carlab LAN6 and Carlab WLAN6, will be implemented using the same physical network and IPv6. NAT and VPN routing are not necessary with the IPv6 networks, as they are globally routable.

Normal development and administration is carried out in the Carlab subnet, protected both from Internet threats and from escalation of breaches into the intranet. Preventing incident escalation by separating Carlab development environment from the intranet is important, because the work includes connecting proof-of-concept and

Name of the network	Subnet	Remarks
Carlab LAN	172.31.104.64/26	For servers and workstations
Carlab WLAN	192.168.1.0/24	For laptops and devices
Carlab VPN	192.168.2.0/24	For remote administration
Carlab LAN6	Public 6to4 (half)	Twin network, IPv6 concept
Carlab WLAN6	Public 6to4 (half)	Twin network, IPv6 concept

Table 9.3: Network subnets in Carlab

Name of the network	Firewall	Authentication
Carlab LAN	firewall, NAT	none
Carlab WLAN	firewall, NAT	Pre-shared key
Carlab LAN6	none	none
Carlab WLAN6	none	Pre-shared key
Carlab VPN	firewall, NAT	VPN authentication

Table 9.4: Network topology in Carlab, security characteristics

demonstration servers to the Internet, which might increase the risk of penetration by malicious attack vectors, and the risk of data theft.

Remote administration of the servers, and working from home is allowed by installing OpenSSH services to all the relevant servers and deploying an OpenVPN router so that Virtual Private Network connections into the Carlab subnet are possible.

The public Carlab servers are accessible from the Internet through NAT router, but actually reside in Carlab subnet. Administration connections to the public servers are not allowed from the Internet, but only from the Carlab LAN subnet.

9.5 Home Domain and Car Domain

The previous sections have concentrated mainly on the Service domain and the Infrastructure domain of the Carbook System. In actual implementation, these two domains are implemented together, because there are no actual third party service providers in the Carlab. All the services are therefore implemented in the same physical network as the Carbook Directory Service. However, the Carbook System has two additional domains, the Home domain, and the Car domain. These domains will be implemented as mock-ups.

The Home domain is implemented by constructing a small living room with an ADSL network connection, inside the Carlab. The Car domain will be built inside a front part of a car, also installed inside the physical space of the Carlab. The

Car domain will be connected to the Internet wirelessly, possibly in a nomadic way, alternating between WLAN and 3G connections.

While there will be only one Home domain in the Carlab demonstration environment, the Car domain is implemented as a group of mobile nodes with relevant software, in addition to the actual Car domain mock-up. These simulated Car domains have different scopes and accuracies of simulation, but they provide the location data that is expecially relevant when assessing Car domain related technologies and system characteristics.

9.6 Next Steps

The general guideline for the Carlab implementations will be aiming towards full Carbook functionality with small incremental steps. However, there are many infrastructural challenges to solve as well. For example, the mail delivery from the servers to administration and users should be made possible by adding relevant mail servers to Carlab.

It is not clear, if it is possible to have native IPv6 connectivity for the mobile terminals. If native IPv6 connectivity cannot be accomplished, the 6to4 method will be assessed next. While the current situation looks rather bleak for IPv6, the transition will inevitably happen, and it pays to be ready for it.

The first demonstrations start in Fall 2008, and at that point we should have a good number of applications and services to build on. The milestones for the Carlab project are set for every year to Spring, Fall and Winter, which makes a good four months of implementation time for every step.

10. CONCLUSIONS

The goal of this thesis work is to define the concept of Carbook and its initial form. Additionally, the technology field relating to the envisioned Carbook framework was mapped and evaluated. These goals were achieved, and it is now possible to design implementation roadmaps with a clear view towards the end result.

Proof-of-concept implementations and demonstrations will be situated in the special Carbook technology laboratory, Carlab. The work will continue with incremental proof-of-concept and technology demonstrations, building small demonstrable sets of services on top of the previous milestone. The growth of the concept platform will be both horizontal, with added low-level services, and vertical, with derived and aggregated services aimed towards end users.

Different options for web platforms with web services capability will need to be evaluated when the actual implementation of the Carbook servers is started. These platforms include, for example, OSGi platform, Apache Axis2 engine, and Web Services Interoperability Technology platform. Tools and frameworks for building web services on top of selected platforms are needed. At the time of writing the Apache Muse looks promising in respect to being compatible with both the OSGi platform and the Apache Axis engine.

Also, a deeper review of the different Web Services specifications in relation to the platforms available, technical capabilities and characteristics of these platforms and the set of features necessary for Carbook System is necessary before the actual implementation of Carbook System. Evaluation of these platforms and additional tools is outside the scope of this thesis.

The choice between supporting UPnP and Devices Profile for Web Services should be addressed with possible later migration plan to DPWS. It will probably be necessary to support both of these UPnP standards depending on the multimedia device availability in the consumer market.

It might be necessary for the Carbook services to serialize and migrate between different platforms. This might be a special platform functionality related to availability, where the services provided by service providers might migrate and replicate inside a cloud computing cluster, or a more interesting case of user agents migrating from one platform to another to for example make use of local services that are not available over the Internet. This, and other agent related technologies will probably gain some extra momentum as the Carbook service platform creates a new market for independent software agents.

It is necessary to conduct some research into the field of software agents to find out what solutions have been found for situations where multiple software agents act hierarchically towards independent goals while all representing the user of the system. The practical problems arising from this are firstly, the potentially conflicting actions and goals between separate agents, and secondly methods of reliably controlling swarms of software agents using intuitive user interfaces.

The Carbook framework aims to be as open as possible without compromising security and stability. Open platforms need an open and active community around them to prosper, and building and supporting the community will become an essential aspect in the future productization of Carbook.

BIBLIOGRAPHY

[1] European Union. Measures to be taken against air pollution by emissions from motor vehicles. DIRECTIVE 70/220/EEC amendment 2002/80/EC [directive].

[2] European Union Sixth Research Framework Programme (FP6). Ambient Networks Project [web site] [referenced 2008-05-30] Available at: http://www.ambient-networks.org/.

[3] International Organization for Standardization. Road vehicles – Communication on FlexRay. ISO/CD 10681 [standard].

[4] International Organization for Standardization. Road vehicles – Diagnostics on Controller Area Networks (CAN) – Part 4: Requirements for emissions-related systems. ISO 15765-4 [standard].

[5] International Organization for Standardization. Road vehicles – Interchange of digital information on electrical connections between towing and towed vehicles. ISO 11992 [standard].

[6] International Organization for Standardization. Road vehicles – Automotive multimedia interface. ISO 22902 [standard].

[7] Institute of Electrical and Electronics Engineers. IEEE 802.21 [standard].

[8] International Organization for Standardization. Road vehicles – Controller area network (CAN). ISO 11898 [standard].

[9] LIN Consortium [web site] [referenced 2008-05-30]. Available at: http://www.lin-subbus.org/.

[10] Internet Engineering Task Force. Mobility Support in IPv6. IETF RFC 3775 [standard] [referenced 2008-05-30]. Available at: http://tools.ietf.org/html/rfc3775.

[11] Internet Engineering Task Force. Fast Handovers for Mobile IPv6. IETF RFC 4068 [standard] [referenced 2008-05-30]. Available at: http://www.ietf.org/rfc/rfc4068.txt.

[12] Internet Engineering Task Force. Network Mobility (NEMO) Basic Support Protocol. IETF RFC 3963 [standard] [referenced 2008-05-30]. Available at: http://www.ietf.org/rfc/rfc3963.txt.

[13] World Wide Web Consortium. Web Services Description Language (WSDL) 1.1 [standard]. W3C Note 15 March 2001 [referenced 2008-05-30]. Available at: http://www.w3.org/TR/wsdl.

[14] World Wide Web Consortium. Web Services Choreography Description Language Version 1.0 [standard]. W3C Candidate Recommendation 9 November 2005 [referenced 2008-05-30]. Available at: http://www.w3.org/TR/ws-cdl-10/.

[15] World Wide Web Consortium. Web Service Semantics - WSDL-S [standard]. W3C Member Submission 7 November 2005 [referenced 2008-05-30]. Available at: http://www.w3.org/Submission/WSDL-S/.

[16] World Wide Web Consortium. Resource Description Framework [standard] [referenced 2008-05-30]. Available at: http://www.w3.org/RDF/.

[17] World Wide Web Consortium. OWL Web Ontology Language Overview [standard] W3C Recommendation 10 February 2004 [referenced 2008-05-30]. Available at: http://www.w3.org/TR/owl-features/.

[18] World Wide Web Consortium. OWL Web Ontology Language Guide [standard] W3C Recommendation 10 February 2004 [referenced 2008-05-30]. Available at http://www.w3.org/TR/owl-guide/

[19] World Wide Web Consortium. OWL Web Ontology Language Reference [standard] W3C Recommendation 10 February 2004 [referenced 2008-05-30]. Available at: http://www.w3.org/TR/owl-ref/.

[20] World Wide Web Consortium. Scalable Vector Graphics (SVG) 1.1 Specification [standard]. W3C Recommendation 14 January 2003 [referenced 2008-05-30]. Available at: http://www.w3.org/TR/SVG11/.

[21] Internet Engineering Task Force. Extensible Messaging and Presence Protocol (XMPP): Core. IETF RFC 3920 [standard] [referenced 2008-05-30]. Available at: http://tools.ietf.org/html/rfc3920.

[22] Internet Engineering Task Force. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence IETF RFC 3921 [standard] [referenced 2008-05-30]. Available at: http://tools.ietf.org/html/rfc3921.

[23] Internet Engineering Task Force. Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP) IETF RFC 5122 [standard] [referenced 2008-05-30]. Available at: http://tools.ietf.org/html/rfc5122.

[24] XMPP Council. XEP-0072: SOAP Over XMPP [standard]. [referenced 2008-05-30]. Available at: http://www.xmpp.org/extensions/xep-0072.html.

[25] OASIS. UDDI Specifications [standard]. [referenced 2008-05-30]. Available at: http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm.

[26] Microsoft Corporation. Devices Profile for Web Services [standard]. February 2006 [referenced 2008-05-30]. Available at: http://schemas.xmlsoap.org/ws/2006/02/devprof/.

[27] UPnP Forum [web site]. [referenced 2008-05-30]. Available at: http://www.upnp.org.

[28] Digital Living Network Alliance [web site]. [referenced 2008-05-30]. Available at: http://www.dlna.org.

[29] IETF WEBDAV Working Group [web site]. [referenced 2008-05-30]. Available at: http://ftp.ics.uci.edu/pub/ietf/webdav/.

[30] IEEE Intelligent Transport Systems Society [web site]. [referenced 2008-07-09]. Available at: http://www.ewh.ieee.org/tc/its/.

[31] Google Inc. Keyhole Marking Language [web site] [referenced 2008-05-30]. Available at: http://code.google.com/apis/kml/.

[32] fmipv6.org. Web site of Linux implementation of IETF RFC 4068 Fast Handovers for Mobile IPv6 [web site] [referenced 2008-05-30]. Available at: http://www.fmipv6.org/.

[33] OSGi Alliance [web site] [referenced 2008-05-30]. Available at: http://www.osgi.org.

[34] OpenStreetMap [web site] [referenced 2008-05-30]. Available at: http://www.openstreetmap.org.

[35] Google Talk for Developers [web site] [referenced 2008-05-30]. Available at: http://code.google.com/apis/talk/open_communications.html

[36] Global System for Telematics [web site] [referenced 2008-05-30]. Available at: http://www.gstforum.org/.

[37] Sun Microsystems. Java technology Sun Developer Network [web site] [referenced 2008-05-30]. Available at: http://java.sun.com/.

[38] Twitter [web site] [referenced 2008-05-30]. Available at: http://twitter.com

[39] Jaiku [web site] [referenced 2008-05-30]. Available at: http://jaiku.com

[40] Symbian Limited. Symbian Signed [web site] [referenced 2008-05-30]. Available at: https://www.symbiansigned.com.

[41] Mark Weiser. Ubiquitous Computing [web site] [referenced 2008-07-09]. Available at: http://www.ubiq.com/hypertext/weiser/UbiHome.html.

[42] Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures [dissertation] [referenced 2008-05-30]. Available at: http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm.

[43] Yoshihiro Ohba - Toshiba America Research, Marc Meylemans - Intel, Subir Das - Telcordia Technologies. Security Signaling During Handovers. IEEE 802 Tutorial [web tutorial]. March 2008 [referenced 2008-05-30]. Available at: http://www.ieee802.org/802_tutorials/march08/21-08-0080-01-0sec-security-signaling-during-handovers-tutorial.ppt.

[44] Sun Microsystems. Online tutorial for Java Archives [web tutorial] [referenced 2008-05-30]. Available at: http://java.sun.com/developer/Books/javaprogramming/JAR/.

[45] Paul Schmitz, Geoff Weaver. MIPv6: New Capabilities for Seamless Roaming. Among Wired, Wireless, and Cellular Networks. DeveloperUPDATEMagazine Intel [e-magazine]. September 2002 [referenced 2008-05-30]. Available at: http://www.intel.com/technology/magazine/communications/nc09024.pdf.

[46] Basavaraj Batil. IP Mobility Ensures Seamless Roaming. Communication Systems Design [e-magazine]. February 2003 [referenced 2008-05-30]. Available at: http://img.cmpnet.com/commsdesign/csd/2003/feb03/feat1-feb03.pdf.

[47] OSGi Alliance. OSGi Provides Open Platform for the Internet-Enabled Car [press release] [referenced 2008-05-30]. Available at: http://www.osgi.org/wiki/uploads/News/pressrel1016900.pdf.

[48] OSGi Alliance. Listener pattern considered harmful [whitepaper] [referenced 2008-05-30]. Available at: http://www.osgi.org/wiki/uploads/Links/whiteboard.pdf.

[49] H. Cervantes and R.S. Hall. OSGi in a nutshell [presentation]. 01 March 2004. [referenced 2008-05-30]. Available at: http://gravity.sourceforge.net/servicebinder/osginutshell.html.

[50] Bill Venners and Elliotte Rusty Harold. Why PUT and DELETE? A Conversation with Elliotte Rusty Harold by Bill Venners [interview] [referenced 2008-05-30]. Available at: http://www.artima.com/lejava/articles/why_put_and_delete.html.

[51] R. Kango, P.R. Moore, J. Pu. Networked smart home appliances - enabling real ubiquitous culture [article]. Networked Appliances, 2002. Liverpool. Proceedings. 2002 IEEE 5th International Workshop 30-31 Oct. 2002 Page(s): 76 - 80 [referenced 2008-05-30]. Available at: http://ieeexplore.ieee.org/iel5/8793/27828/01241340.pdf.

[52] Sanjiva Weerawarana. Google Tech Talks Web Services Middleware: All Grown Up! [speech]. 2007-10-08. [referenced 2008-05-30]. Available at: http://video.google.com/videoplay?docid=4366223572258324894.

[53] Matt Tucker. Jive Talks XMPP (a.k.a. Jabber) is the future for cloud services [blog]. 2008-01-24. [referenced 2008-05-30]. Available at: http://www.jivesoftware.com/community/blogs/jivetalks/2008/01/24/xmpp-aka-jabber-is-the-future-for-cloud-services.

[54] Tom Jordahl. SOAP over XMPP [blog]. 2005-09-06. [referenced 2008-05-30]. Available at: http://tjordahl.blogspot.com/2005/09/soap-over-xmpp.html.