NTNU
Innovation and Creativity

Mobile Home Access
with Tunneling Support

Tan Nguyen

Master of Science in Communication Technology
Submission date: July 2006
Supervisor: Van Thanh Do, ITEM
Co-supervisor: Ivar Jørstad, ITEM

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Lately, there is an explosion in the usage of personal computers and equipment at home. More and more households have PCs connected to the Internet via ADSL, Cable TV, etc. Several applications such as email, web browsing, chat, etc. have become popular. Private files such as documents, pictures, music, etc. are stored in Home PCs. More advanced home users even have a Home Local Area Network that connects together peripheral devices like printer, scanner, camera, microphone, etc., and also consumer electronics like refrigerators, oven, alarm system, electricity control, etc. Briefly, the User Home System is getting more and more advanced. The need to access the User Home System from mobile phones, external computers or other embedded and connected devices will soon be a reality. The goal of the Mobile Home Access is to anticipate such an evolution. It will propose a solution that enables mobile users personalized access to his/her Home System and services, and support for new terminal types should be easy to add as they emerge (e.g. handheld mp3-players that have Internet access). Two alternative solutions are identified. The reduced mapping solution is more suitable for mobile devices with limited network resources and processing power, while the tunneling solution is intended for regular PCs and laptop computers.

The thesis work consists of the following tasks:

- Requirements Analysis and Design of the tunneling solution for the Mobile Home Access
- Design and implementation of a Tunneling Home Access XML Web Service
- Design and implementation of a Tunneling Home Access WS client for PCs

Assignment given: 09. February 2006
Supervisor: Van Thanh Do, ITEM

Abstract

Mobile phones and electronic gadgets on the market today have become more and more powerful, in both processing power and functionalities. Accessing files and documents residing at users' home networks via a vast amount of devices is therefore anticipated.

The current solution to remote home access, i.e. Virtual Private Network (VPN), is not supported on all different types of devices. In particular, limited processing power and memory footprint on mobile devices such as PDAs and cell phones are not suitable for VPN clients that require high processing power during encryption and decryption of data. In addition, setting up VPN on home network is not a straight-forward task that anyone can complete. Regular users do not have knowledge to set up and configure VPN correctly.

It is obviously that we need another approach/method to access home network in an easier manner. The specified system, Mobile Home Access, will provide ubiquitous access to the home network independent of network infrastructures, platforms and terminals. This goal has many challenges and obstacles, such as firewall and routing issues as well as compatibility and the restrictions of the well-known networked file system, i.e. Common Internet File System (CIFS).

Based on the previous research project, Mobile Home Access will be implemented as web services and written in Java, the platform independent programming language. XML web services offers intercommunication between applications and protocols running on various network infrastructure. Using web services, the networked file system's services can be exposed to remote clients.

The system will operate in two different modes, the reduced-mapping mode for restricted clients (mobile phones, PDA, etc) and the tunneling mode for rich clients (desktop computers, laptop, etc). This thesis will address and try to overcome challenges associated with the tunneling mode in addition to the design of the overall system.

Acknowledgement

This thesis is carried out at Norwegian University of Science and Technology (NTNU) in spring semester 2006 and originates as part of an earlier research project. The working place and lab are at Telenor Technology Centre in Fornebu.

I would like to thank my “colleagues” working at Telenor Technology Centre for a great and cheerful time in the working lab. Special thanks to Mr. Ivar Jørstad and professor Do van Thanh for their valuable comments, encouragement and support. Without them, my thesis would never have been completed.

July 28, 2006 - Fornebu

Nguyen van Tan

Table of contents

List of figures

List of tables

Abbreviation

AD
Active Directory
API
Application Programmer Interface
ARP
Address Resolution Protocol
BTS
Base Transceiver Station
BDC
Backup Domain Controller
CDC
Connected Device Configuration
CIFS
Common Internet File System
CLDC
Connected Limited Device Configuration
CORBA
Common Object Request Broker Architecture
DHCP
Dynamic Host Configuration Protocol
DNS
Domain Name Service
FTP
File Transfer Protocol
GPRS
General Packet Radio Service
GUI
Graphical User Interface
HTTP
Hypertext Markup Language
IDL
Interface Description Language
IP
Internet Protocol
IPC
Inter-Process Communication
ISP
Internet Service Provider
J2EE
Java 2 Enterprise Edition
J2ME
Java 2 Micro Edition
J2SE
Java 2 Standard Edition
JAR
Java Archive
JDIC
Java Desktop Integration Components
JDK
Java Development Kit
JNI
Java Native Interface
JRE
Java Runtime Environment
JSR
Java Source Request
JVM
Java Virtual Machine
LAN
Local Area Network
MIDP
Mobile Information Device Profile
MIME
Multipurpose Internet Mail Extensions
NAPT
Network Address Port Translator
NAT
Network Address Translator
NBNS
NetBIOS Name Service
NBT
NetBIOS over TCP/IP
P2P
Peer-to-peer
PC
Personal Computer
PDA
Personal Digital Assistance
PDC
Primary Domain Controller
RFC
Request for Comments
RPC
Remote Procedure Call
RUP
Rational Unified Process
RMC
Rational Method Composer
SMB
Server Message Block
SMTP
Simple Mail Transfer Protocol

SAAJ
Soap with Attachment API for Java
SOA
Service-Oriented Architecture
SOAP
Simple Object Access Protocol
SSH
Secure Shell
TCP
Transport Control Protocol
UDDI
Universal Description, Discovery, and Integration
UDP
User Datagram Protocol
UML
Unified Modeling Language
UMTS
Universal Mobile Telecommunications System
UNC
Universal Naming Convention
URI
Uniform Resource Identifier
URL
Uniform Resource Locator
USB
Universal Serial Bus
VPN
Virtual Private Network
WAR
(Java) Web Archive
WINS
Windows Internet Naming Service
WLAN
Wireless LAN
WSDL
Web Service Description Language
WWW
World Wide Web
XML
Extensible Markup Language
XSD
XML Schema Definition

Chapter 1 Introduction

Today, our lives and our way of living has been evolved and value-added by computers and small electronic devices. For example mobile phones, mp3 players, digital cameras are more and more common in a typical Norwegian household and also around the globe. The number of households with Internet access via broadband is also increasing every day. Statistically, sixty-four percent of the Norwegian household have access to Internet, and there were 778 000 broadband subscriptions in the first quarter of 2005 [1].

The home networks have become more advanced, and may include consumer electronic devices such as refrigerators, ovens, and alarm systems. The idea of remotely accessing our files, documents and resources at home is not a new discovery or need, but rather no implementation exist that meet the future's need of remote home access from a vast amount of devices in addition to new and better devices arriving to the market every day

As technologies are evolving at this rate, we anticipate that users will need to access their home networks from their handheld devices and all other types of terminals at many different locations via an Internet connection.

1.1 Definition & Objectives

Definition of Mobile Home Access:

- **Mobile**
  The term *Mobile* is not only associated with mobile phones or devices, but also with different types of mobility, such as terminal mobility and personal mobility[1]. *Mobile* also means that users could be located anywhere and on the move from one location to another.
- **Home**
  The term *Home* refers to user's private home network. It also includes all services, resources and devices that participate in this network.
- **Access**
  The term *Access* means that users can read, modify and execute available services or resources.

Thus, Mobile Home Access is a system that provides ubiquitous access to home network from a vast amount of electronic devices.

A picture tells more than thousand words. The objectives of Mobile Home Access system are depicted in the figure 1.1 below. There are simplicities in this figure:

1. The home network consists of two computers, *CompA* and *CompB*. In fact, the home network is more advanced and may include wired and wireless networks as well as different types of networked devices (scanner, printer, consumer electronic devices, laptop, WLAN-enabled PDA or other mobile devices).
2. Only a mobile phone and desktop computer are shown as clients, but remote clients could be all types of electronic devices with Internet access.

[1] Terminal mobility allows the terminals to change location while maintaining all services. Personal mobility allows users to access all services independently of terminals and networks [2].

3. The red and blue lines indicate possible data flow between mobile devices and the home network, and between remote computers and the home network, respectively. In reality, the route for data through Internet changes dynamically all the time and is unpredictable. This is by design of Internet.

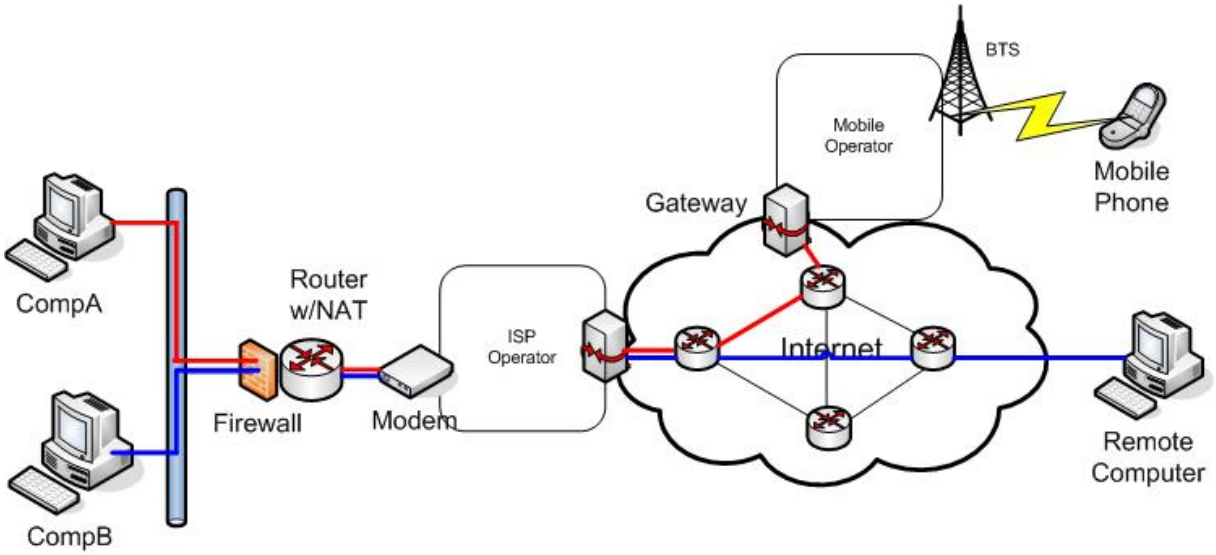

Figure 1.1 Mobile Home Access

1.2 Problem Statements

The main goal is to design and implement the Mobile Home Access system that provides users with ubiquitous and transparent access to their home network from various types of remote terminals including stationary and mobile handheld devices. In addition to satisfy the main goal of the system, the system should improve the shortcomings (of existing solutions) and overcome all challenges related to the exposure of the networked file system.

To achieve Mobile Home Access's objectives, the specified system must overcome many challenges and problems of nowadays restrictions.

Firewall

Firewalls reside at the user's private home network, these include personal firewalls which filter traffic entering and leaving a single computer and firewall in the router. The main purpose of a firewall is to block traffic that is considered insecure and originated outside the internal network. The networked file system, CIFS, is intended for internal network and the security is weak. The resources are sometimes even open, without any kind of protection such as using passwords, for everyone inside the network. Obviously, the firewall will block requests to the networked file system, i.e. at the port numbers 137, 138, 139 and 445.

The challenge is how to bypass the firewall for requests to the internal CIFS servers in the home network.

Routing and Network Address Translator (NAT)

Usually the Internet Service Provider (ISP) offers one public IP address for a broadband subscription, but the home networks tend to consist of many computers and terminals. The home networks use private IP addresses[2] to assign to computers and nodes connected to the

[2] Private IP address range [3]: 10/8, 172.16/12, 192.168/16

router. The use of private IP address is a technique to overcome the shortcomings of available IPv4 addresses. Unfortunately, computers and their services inside the private networks are not accessible outside the home network unless it is configured in the router to forward traffic that meets a predefined condition (such as port number). This also limits the number of public available services at a given port to exactly one.

The challenge is how to route traffic to the computer hosting the requested resources and files for the CIFS service at the same port numbers, i.e., all computers on the home network must be reachable through one endpoint defined by an IP-address/port-number pair.

Dynamic IP Allocation
The IP address supplied by ISP is either static or dynamic. In case of dynamic IP addresses, the IP address of the home network may not be known ahead in time. This is due to the design of an ISP, because their number of customers exceeds the number of available IP addresses. The ISP's Dynamic Host Configuration Protocol (DHCP) server automatically assigns a free IP address to the client. This IP address will expire at some period of time and the client is forced to query the DHCP server for new IP address. It is even worse for users with dial-up connections, because the IP address changes for each dial-up connection.

The challenge is how to learn the public IP address of the home network. Remote clients must be able to discover/find the IP address of their home network. The Mobile Home Access must also notify its clients when IP address changes.

Device/Terminal Limitations
Every single device is designed to meet certain functionalities, for example mobile phones to enable us to make and receive phone calls wherever in the world. Such devices have many restrictions and limitations for support of Internet access.

The challenge is to enable remote access for these devices and for new devices arriving on the market in the future. This challenge is one of the main reasons why Mobile Home Access is brought to life, because we want to access our files for all types of devices with Internet access as well as restricted terminal/computers independent of their platforms, network architectures.

Virtual Network Interface
On remote clients the traffic destined to home network must be forwarded to the Mobile Home Access clients. This is done by using a virtual network interface. The challenge is to implement such functionality.

Note:
From the research project [4] we have found "theoretic" solutions for many of the challenges mentioned above. XML Web Services as solution to bypass the firewall and Java to overcome the challenges in terminals. The Mobile Home Access server will handle the routing problem by acting as a proxy for internal services.

1.3 The Scope of the Thesis

Mobile Home Access has been identified to work in two different modes, Reduced Mapping Mode for restricted clients and Tunneling Mode for rich clients. This thesis will design and implement the Tunneling Mode, in addition to look at the design of the overall system.

Security functions for Mobile Home Access, such as authentication, authorization, and encryption/decryption will not be implemented. However, a simple authentication mechanism for login is implemented. Throughout this document, it will mention where (stronger) security mechanisms are required/recommended to protect user's privacy.

Technologies related to the Reduced Mapping Mode and/or irrelevant to Tunneling Mode will not be mentioned.

1.4 Organization of the Thesis

This is a short overview of the chapters' content in this thesis. A reader's guide is presented in figure 1.2. This thesis is mainly divided into three parts:

- *Part I: Basic Concepts*
  presents architecture, protocol details and key technologies that Mobile Home Access is based on.
- *Part II: Design*
  consists of design and analysis of the Mobile Home Access
- *Part III: Implementation*
  includes implementation detail and experiences
- *Part IV: Evaluation*
  consists of testing and performance evaluation

Chapter 2 Related Works
Examines technologies and previous works that are related to the specified system. This includes the Virtual Private Network and Peer-2-Peer File Sharing.

Chapter 3 Methodology
Describes the project's methodology. A short overview of Rational Unified Process (RUP) as the main workflow of this thesis is provided.

Part I Basic Concepts
Chapter 4 Java
Provides an overview of the Java programming language and relevant Java APIs.

Chapter 5 Networking
This chapter describes briefly the two most common network architectures, the peer-to-peer and client-server (domain) network architecture. Network Address Translator and Firewall technologies are also described.

Chapter 6 CIFS
CIFS is a networked file system and it is the core protocol for our services. This chapter also discusses protocols related to CIFS which are NetBIOS and the Unix version of CIFS, Samba.

Chapter 7 XML Web Services
XML Web Services is an implementation of the Service-Oriented Architecture (SOA). This chapter discusses SOAP, WSDL and UDDI.

Part II Design
Chapter 8 Requirements
In this chapter, the use cases will be identified and then used to construct requirements of the system's requirements.

Chapter 9 System Architecture
Architecture of the Mobile Home Access is described in this chapter.

Chapter 10 System Design & Analysis
The design of Mobile Home Access depicted with UML diagrams.

Part III Implementation
Chapter 11 Development Environment & Tools
Environment and tools in the development.

Chapter 12 MHA Server
Describes the implementation of Mobile Home Access server including web services for Tunneling Mode and other issues.

Chapter 13 MHA Client
Concepts and implementation of virtual network interface and web service invocation interface.

Chapter 14 Experiences
Author's experiences and pitfalls during development process.

Part IV Evaluation
Chapter 15 Test
Introduces JUnit as the testing framework.

Chapter 16 Performance
How performance can be calculated.

Chapter 17 Validation
List the requirements and how well it is achieved.

Chapter 18 Future Works
Author's recommendations of works for future development of Mobile Home Access.

Chapter 19 Conclusion
This thesis' conclusion as short-term and long-term impact of Mobile Home Access.

Below is a figure that shows how chapters are organized and the order the author recommends to reading them.

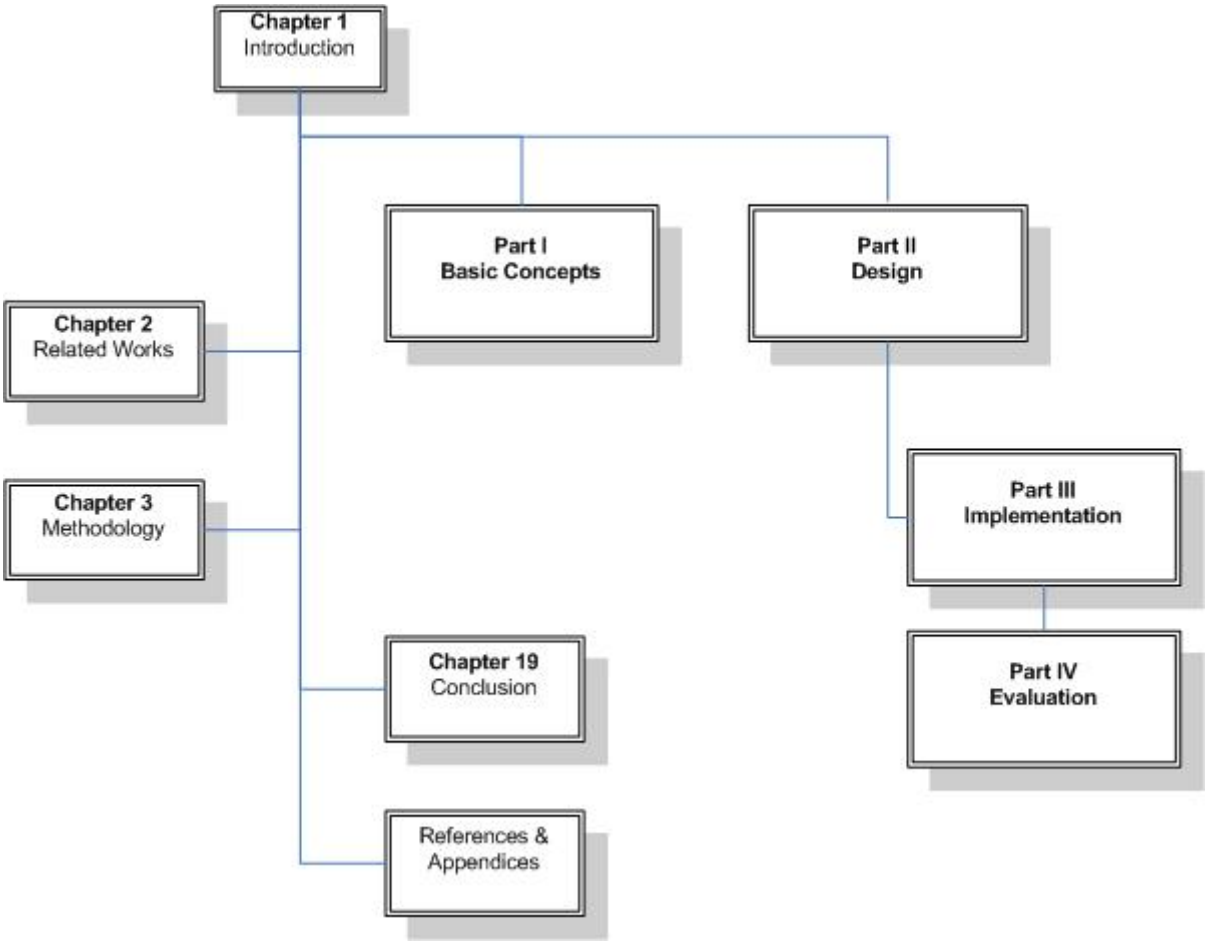

Figure 1.2 Reading Guide

References and *Appendices* can be found at the end of this document. All *source codes* and Java API are available online at http://mha.tanvn.com/.

Chapter 2 Related Works

Understanding the objectives and requirements of Mobile Home Access is quite important, as well as investigating if this is already been solved or implemented. At this point we will decide if our project will continue to the design phase. Related works are great sources to our project as we investigate what objectives and limitations they have. Many open source projects may have implemented similar features as Mobile Home Access. In that case, we can reuse some of their source code. This approach is becoming more and more common in software development as it provides low time-to-market and high quality codes that have been well tested. Reusability is one of the keywords in modern programming practice. As a result, Mobile Home Access will be designed with high degree of reusability and extensibility in mind.

There are two technologies, Virtual Private Network and peer-to-peer (P2P) file sharing, that enable remote users to access their files at home network. Unfortunately, none of them satisfies the problem domain and objectives of Mobile Home Access. We will discuss their objectives and the limitations that make them not suitable in our case.

2.1 VPN

A Virtual Private Network (VPN) is a network that is constructed by using public telecommunication infrastructure, i.e. the Internet, to connect remote sites and users together. It was first designed to give companies the same capabilities as private leased lines at much lower cost. VPN is now a mature technology and de-facto standard used by many companies.

Definition of VPN:

Virtual
Users access the services at the remote network as though the services are on the local network and get the impression as they are physically connected to the same LAN.

Private
VPN maintains privacy using tunneling protocols and security procedures, and ensures the confidentiality and integrity of the data as well as user authentication.

Network
VPN combines and interconnect multiple LANs from different locations. This enables companies to exchange data and services securely between their offices in addition to customers, collaborating partners and contractors.

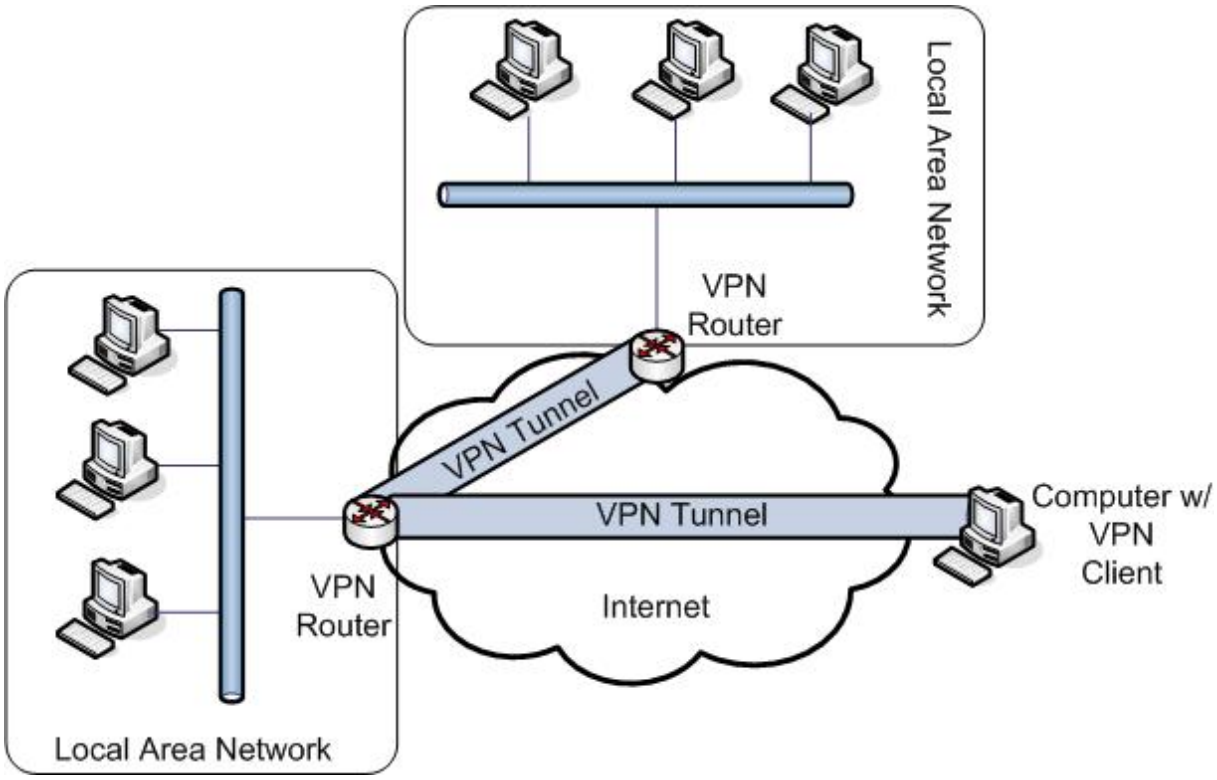

Figure 2.1 Virtual Private Network

As in figure 2.1, two Local Area Networks are connected via VPN-enabled routers. Computers at one LAN can now communicate with computers on the other LAN in the same way as communicating between two computers on the same LAN. Data are encrypted and decrypted in the router thus create a virtual tunnel, and ensure that data cannot be intercepted. In this way, VPN provides a secured network infrastructure. Remote clients, with a VPN client installed, can connect to the network and access internal services but this time a VPN tunnel is created between remote client and VPN Router (as shown in figure above).

VPN is fairly close to mobile home access, but it has limitations that make it unsuitable for our requirements. These limitations are:

- VPN is designed for computers, and is not suitable for mobile and restricted devices. Thus, VPN *is not terminal independent*.
- VPN is *difficult to setup* for regular users. The procedures of setup and configuration require advanced knowledge and there are many pitfalls users can run into.
- VPN client application is "big" in *size* which results in *high memory footprint* and uses/installs a (virtual) network interface. Today's mobile devices have restricted resources and do not have such capacity and functionality.
- VPN security functions such as encryption and decryption *require high processing power*, which mobile devices cannot achieve.

2.2 P2P File Sharing

A peer-to-peer network is a network where each node uses its own processing power and bandwidth instead of concentrating on server(s). Each node is equal and functions as client and server at the same time. Such networks are useful for many purposes, but most popular is for file-sharing such as Gnutella [5]. In fact, Windows Workgroup is such a network where

every computer shares its resources to others (server) and access services/resources provided by other computers (client).

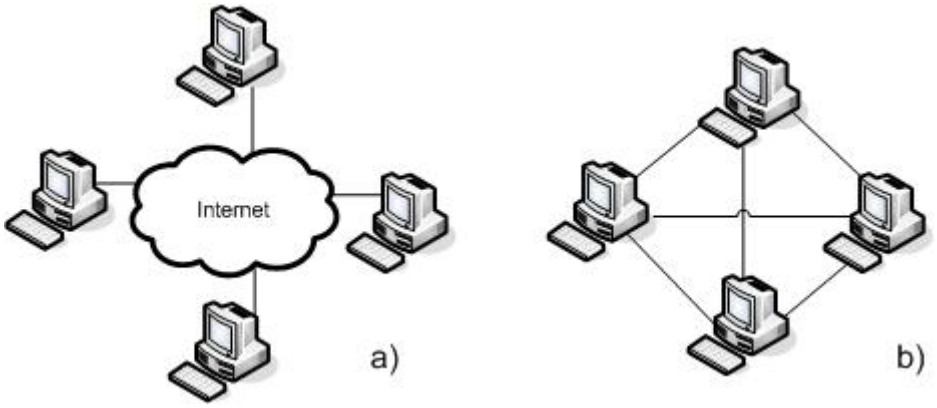

Figure 2.2 P2P a) Physical representation b) Logical representation

The P2P network for file sharing satisfies the requirements of accessing and sharing files and documents in the home network but lacks functionalities of a networked file system. The functionalities that are not available or not implemented by peer-to-peer file sharing networks are:

- *Access services and resources in home network*
  shared files and documents must be in the specified path to enable sharing. Services cannot be shared, for example printing.
- *File operations*
  Operations directly on files and directories, such as open, editing or deletion are not implemented.
- *Access control*
  There is no access control mechanisms (authentication, authorization) on files and folder
- *Mapping/mounting remote resources to local file system*
  It is not possible to mount remote resource as local resource.

Chapter 3 Methodology

Before diving into the system design and implementation, we will describe the methodology that is used in this thesis. It is crucial for the success of any project that the methodology is clearly understood and followed closely to maintain the project's objectives in time. Experiences from the development process shows that a system which is developed using a well-defined methodology requires longer in modeling, but the maintenance and further development of the software is much easier and faster. The software development process is a systematic process that enables us to model and analyze the system in small chunks or abstractions at a time. This helps to reduce and discover the risks early in the development process.

The development process that is used throughout this thesis is based on Rational Unified Process (RUP) [6, 7, 8] developed by Rational Software Corporation, now a division of IBM. RUP is developed on top of experiences in the software industries development and uses Unified Modeling Language (UML), which is a modeling language for specifying, visualizing, constructing, and documenting software systems. RUP is included in IBM Rational Method Composer (RMC), a software development process platform.

3.1 Rational Unified Process (RUP)

The Rational Unified Process is an iterative software development process. RUP is an adaptable process framework in the sense of selecting the development processes from a large number of different activities appropriate to a particular software project.

RUP is based on best practices of modern software development as a way to reduce the risk inherent in developing new software. A RUP-based project's lifecycles are broken into phases. Each phase has objectives to be accomplished using one or more iterations. This is represented in a two-dimensional graph below.

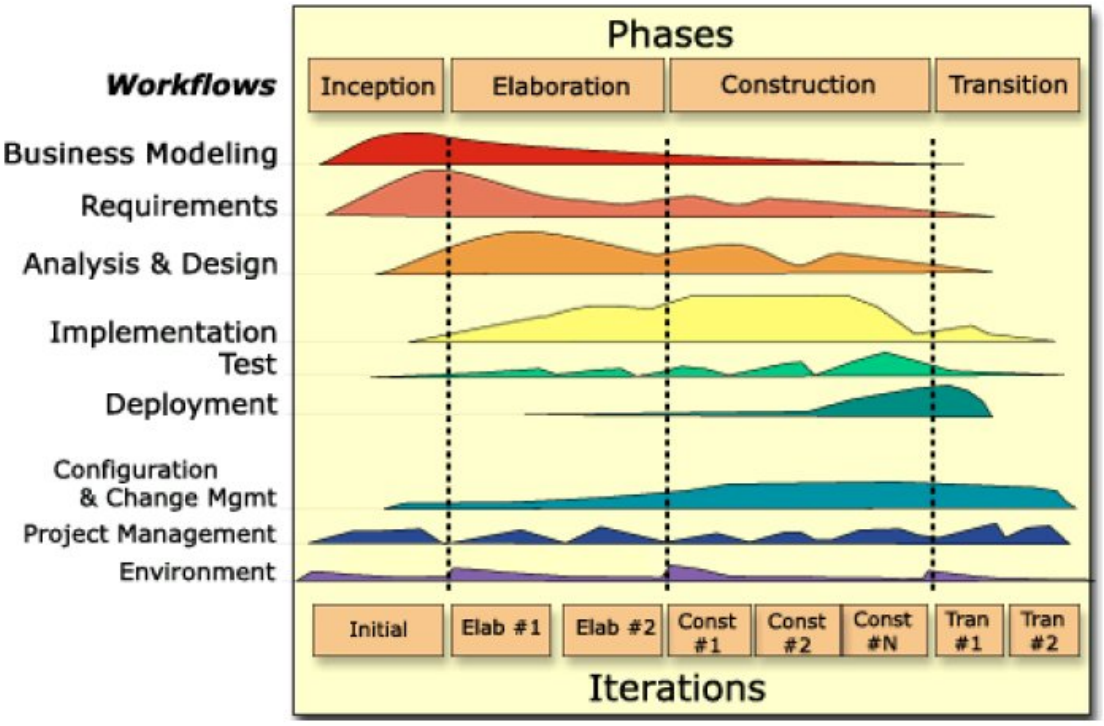

Figure 3.1 RUP Overview

3.1.1 Phases

The RUP process framework has four distinct phases. There is a *milestone* with a set of criteria to be accomplished at the end of each phase before we can move to the next phase. If all the objectives are not satisfied, it would be necessary to perform more iteration or probably a redesign of the software architecture.

Inception Phase

The inception phase is important for new projects, where we must address requirement risks and business cases before the project can proceed.

Typical activities in this phase are:

- Formulate project scope, requirements, constraints
- Use-case model that identifies use cases and actors with little or no implementation details.
- Develop an initial project plan (with phases and iterations)
- Research and investigate potential architectures
- Build or prepare project environment

During inception phase, a project's objective or *vision* is produced. This is a short overview of the system, its features, why it will be used, and who will use it.

Elaboration Phase

In the elaboration phase, the problem domain analysis is made to provide a stable basis for the Construction phase. The main tasks of this phase are to establish architecture foundation of the system and create models that describe the problems to be solved.

Typical activities:

- Supplementary requirements capturing non-functional requirements

- Design the software architecture by developing a use-case model
- Develop an executable architecture that realizes use cases
- Analysis of the software architecture
- Development plan for the overall project

Construction Phase
The main goal of this phase is to complete the implementation of the specified system. During this phase, all components and features are developed and integrated into the product. Features should also be thoroughly tested.

Typical activities:
- Develop components
- Integrate and merge results from concurrent activities
- Testing
- Refactoring

Transition Phase
In the transition phase, we ensure that the software is ready for end-users. This includes all issues required to place the software into the hands of end users.

Typical Activities:
- Finalize end-user support material (deployment plan, release notes, documentation, guides/help)
- Test the product in a customer environment
- Fine tuning the product based upon feedback
- Deliver the final product to the end user

3.1.2 Iterations

Each phase in RUP is broken down into iterations. During each iteration, activities from several diciplines[3] in varying levels of details are performed. Thus, an iteration is a complete development loop. From iteration to iteration, the product grows incrementally until the required objectives of current phase are met. Unlike phases which have objectives, iterations have deadlines.

RUP in essential has the following core workflows; Requirements, Analysis and Design, Implementation, and Test. In fact, RUP has nine core process workflows which represent partitioning of all workers[4] and activities[5].

The core process workflows are divided into six core “engineering” workflows:
1. Business modeling
2. Requirements
3. Analysis & Design
4. Implementation
5. Test
6. Deployment

[3] Disciplines – focus areas of software engineering such as Requirements, Design and Analysis, Implementation, and Test.
[4] A *worker* defines the behavior and responsibilities of an individual or a group/team
[5] An *activity* is a unit of work that an individual in that role may be asked to perform

And three core "supporting" workflows:

1. Project Management
2. Configuration and Change Management
3. Environment

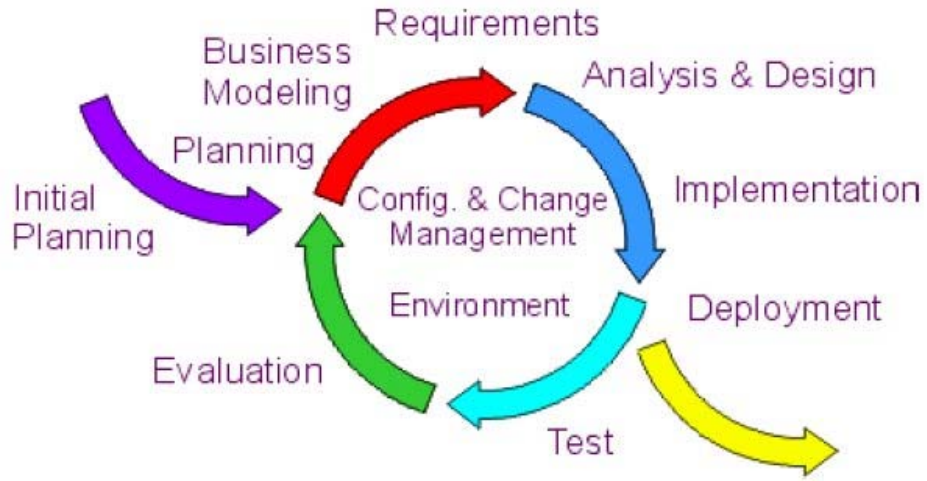

Figure 3.2 Typical Iteration

Requirements

The goal is to describe what the system should do by documenting required functionality and constraints. In projects where the objectives are to improve an existing software application, users of the existing system should participate to identify the problems they may experience as well as functionalities they wish to be implemented.

Following documents should be produced:

- System's requirements, (functional, non-functional and implementation requirements)
- Problems and challenges
- A Vision document (the project's objectives)
- Project Plan

UML diagrams used in this step:

- Use-case model
  UML use case diagrams are commonly used to identify requirements of a system.

Design and Analysis

This step involves translating requirements into a software architecture and show how the system will be realized in the implementation phase. The design model (which consists of design classes structured into packages and subsystems with well-defined interfaces) serves as an abstraction of the source code.

UML diagrams used in this step:

- Sequential diagram
- Interaction diagram
- Class diagram
- Component diagram

Design Patterns
A design pattern is a common solution or approach to solve problems that appear often. Design patterns help to design system easier and faster.

Implementation
Implement components from the design model in Design and Analysis step and test the developed components as units.

Test
The purpose of testing is to ensure that the required behaviors of the system are correct and that all required behaviors are present as specified.

3.1.3 MHA's Intial Project Plan

The iterations for each phase and amount of time they occupy for this thesis's development process is as follow:

- **Inception**
  - Requirements (80%)
  - Analysis (20%)
- **Elaboration**
  - Requirements (40%)
  - Analysis (40%)
  - Design (20%)
- **Construction**
  - Requirements (3%)
  - Analysis (2%)
  - Design (35%)
  - Implementation (50%)
  - Test (10%)
- **Transition**
  - Test (10%)
  - Deployment (90%)

PART I:

BASIC CONCEPTS

Chapter 4 Java

Java programming language [9, 10] enables software application developers to implement applications and users to run them on multiple platforms. It is common to choose the language of programming language at the beginning of the project. Sometimes it is not even an option, because the system is dependent on legacy code/system and the language has better advantages than others do for the specific system.

This chapter describes benefits of Java as the project's language of choice and as a general-purpose programming language. Lately, Java has obtained popularity in application development on mobile embedded devices in addition to desktop and enterprise computers. One of the project's objectives is to develop a system that works on multiple platforms and on a vast amount of devices. Java is widely supported on most consumer electronic devices on the market, which is the main argumentation why to choose Java as the implementation language. Java is ideal for developing secure, distributed, network-based end-user applications in network-embedded devices, Internet and desktop environments.

4.1 Java Fundamentals

Regular users have heard, at least once, Java mentioned when using a service for example in software applications and/or games on mobile phones. Unfortunately, not many do really know what Java actually is and explore its full potential. Probably users may never need to find out anything about Java unless knowing their devices support Java. On PC, if Java is not pre-installed, users must manually (or automatically by the application itself) install it to be able to run Java applets and applications.

The Java programming language and environment is designed by James Gosling and others at Sun Microsystems to solve common problems arising in modern programming languages. The main goal is to meet the challenges of application development in heterogeneous, networked-wide distributed environment. To make it easy to learn and make it familiar to programmers, Java is designed to be as close to C++ as possible. Programmers can grasp the language quickly, concentrate on innovative development, and be productive from the very beginning.

What is Java?

"Java: A simple, object-oriented, networked-savvy, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dynamic language."

The table below shows a short overview and explanation of features and benefits of Java.

Java is …	
Simple	- Automatic garbage collection to overcome problems with allocation and freeing memory - Omits confusion and complex features from its C/C++ ancestors such as operator overloading, multiple inheritance - access built-in libraries, which provide commonly use functionalities and easy to extend

Java is …	
Object-oriented	- Focus on objects and their operations and interfaces - Reuse existing components and libraries - Structural, cleaner but still handle increasing complexity
Network-savvy	- Built-in libraries of routines for networking tasks - Easier to modify network connection similar to read and write local files
Robust	- Strongly typed language, to eliminate error early - extensive compile-time and run-time checking - Pointer model to avoid overwriting and corrupting data
Architecture neutral	- Compiler generates machine- and platform independent byte codes - solve binary code distribution problem and version problem
Secure	- Security sandbox - Byte zed codes
Portable	- Architecture neutral - No system implementation dependency, for example the size of the data type *double* is always the same for all architecture
Interpreted	- The link phase of a program is simple, incremental, and lightweight
High-performance	- Interpreter runs at full speed without checking the run-time environment - Automatic garbage collector runs as low-priority to ensure that memory is available when required
Multithreaded	- Reflects the real world and increase real-time behavior and low response time - Built-in synchronization mechanism
Dynamic	- Dynamic class loader to link only classes as needed and on demand

Table 4.1 Java characteristics

Misconceptions and disadvantages

Compile once, run anywhere

The promise to "Compile once, run anywhere" of Java, which is to compile source code into architecture neutral byte codes that could be understood by all implementation of Java Virtual Machine (JVM) will not work in all cases. In fact, byte codes implicitly requires libraries used in compilation to be supported in the running platform. Unfortunately, not all implementations of JVM support the standards specified in the Java specification, such as the J2ME platform which only support a subset of the standard Java API. This implies that the codes, which are design to work on many different targets, must take this into account. If the application also uses optional APIs, we must also check that this is supported or added them manually. Thus, an application written in Java is not ensured to be able to run properly even when Java is installed.

Performance
Many have argued the speed and performance of Java to be poor and slow in comparison to other languages, such as C/C++. This is true in some cases where the code directly accesses hardware and native code in the operating system. But Java has been improved since the day it was born, and other improvements in speed and performance of the Java language are under development.

4.2 Java on Devices (J2ME)

Devices (such as mobile phones and PDAs) have restricted resources and therefore require the JVM to be small and create small footprint in terms of memory usage and processing power. In addition, some of the classes in Java API are omitted. There are vast amounts of different types of devices that have different capacities, therefore there are a set of configurations and profiles that will meet the specific device.

Configuration
A configuration identifies a family of similar devices and defines a minimum set of JVM and Java library classes. To reduce the number of different implementations of JVM, only two configurations are defined:

- CLDC (Connected Limited Device Configuration)
  For devices with limited resources, such as mobile phones.

- CDC (Connected Device Configuration)
  This configuration is for devices with high resources and processing power, such as PDAs.

Profile
A profile extends the configurations by adding extra class libraries specific for a group of devices. For a configuration, there could be many profiles. Device manufacturer can provide extra functionality by adding libraries. The architecture is extensible and very flexible that allow future devices.

- MIDP (Mobile Information Device Profile) /CLDC
- Information Module
- Foundation Profile/CDC
- Personal Profile/CDC
- Personal Basis Profile

4.3 Java on PCs (J2SE, J2EE)

For PCs, there exist no restriction on processing power and memory and therefore a full version of JVM is installed. For regular users, the Java 2 Standard Edition (J2SE) is sufficient. It includes the JVM, JRE and libraries. For enterprises, Java 2 Enterprise Edition (J2EE) can be used. It includes everything in J2SE plus other packages.

4.4 Relevant Java APIs

In addition to packages or APIs that come with Java platform, many optional APIs can be used in this thesis. Short overviews of these APIs are described below, as well as relevant Java APIs.

4.4.1 Java Native Interface

Java Native Interface (JNI) [11, 12] allows Java code to interact with applications and libraries written in other languages, such as C, C++ and assembly. This native programming interface for Java is part of the Java Development Kit (JDK). By writing programs using JNI, the code will be completely portable across all platforms.

The reasons to use JNI are to handle situations when the applications cannot be written entirely in Java:

- the features needed by the application is not supported
- some features are best handled outside the Java environment
- access existing libraries or applications in another programming language

The JNI framework does not only enable Java applications to create, update, access and share objects with other applications but also enables other applications to interact with Java objects. This is depicted below, respectively.

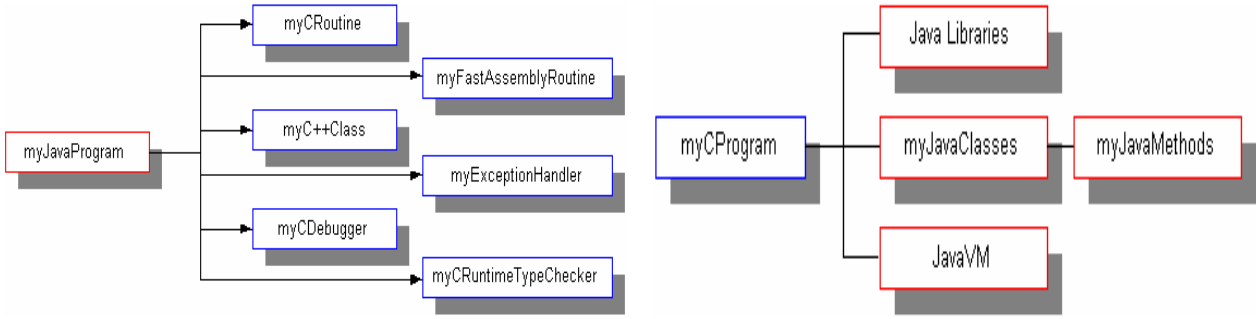

Figure 4.1 a) Java to C++ application b) C++ to Java application

This simple and typical "Hello World" example below shows us how to use the JNI framework. Writing the Java application with JNI includes the following steps:

- Declare a method as native, i.e. implemented by other language
- Load the library. The name of the library is converted to a platform-specific name to the native library.
- The native method is called in the Java class similar to other methods within the class.

```
class HelloWorld {
    public native void displayHelloWorld();

    static {
        System.loadLibrary("hello");
    }

    public static void main(String[] args) {
        new HelloWorld().displayHelloWorld();
    }
}
```

Table 4.2 Java file, HelloWorld.java

The next step is to write the implementation of the native method. This includes:

- Compile the Java file and then generate header file by running *javah -jni* on the Java class file.

- Write the implementation and include the generated header file and JNI header file.

```
#include <jni.h>
#include "HelloWorld.h"
#include <stdio.h>

JNIEXPORT void JNICALL
Java_HelloWorld_displayHelloWorld(JNIEnv *env, jobject obj)
{
    printf("Hello world!\n");
    return;
}
```

Table 4.3 Implementation file, HelloWorldImp.c

The last step is to create the shared library.

```
cc -G -I/usr/local/java/include -I/usr/local/java/include/solaris \
      HelloWorldImp.c -o libhello.so

cl -Ic:\java\include -Ic:\java\include\win32
      -LD HelloWorldImp.c -Fehello.dll
```

Table 4.4 Create the library

JNI API is used in the this thesis to provide our Java application to access network interface information and functionality. Because this is only an overview of JNI, advanced topics about JNI is out of the scope of this thesis.

4.4.2 Java Desktop Integration Components

Java Desktop Integration Components (JDIC) [13] is an open-source project with the goal to provide desktop functionalities for Java application while maintaining cross-platform support. Java developers have before lacked this functionality and had to write their own implementation by using Java Native Interface (as described above). JDIC API is designed to fill the gap so developers do not need to deal with native programming and stick to pure Java only.

JDIC supports the following features in addition to components under construction, which are called *incubator* projects:

- File explorer
- Web browser
- System tray
- Native file support (e.g. open a file with the associated application to the file type)

To enable Java applications to dynamically figure out which operating system it is being run on and to call the appropriate JNI function, JDIC uses a wrapper. This is done in JDIC Java Component Code.

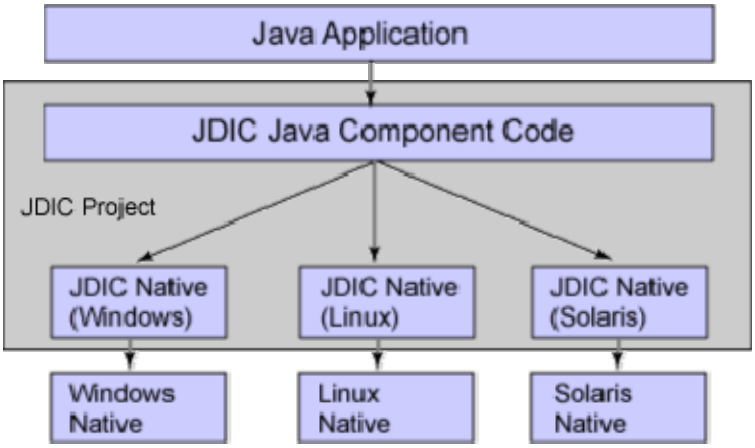

Figure 4.2 JDIC Wrapper

Even though the goal is to enable desktop support for all operating systems that are in use by people, unfortunately JDIC only support Windows, Unix and Solaris at the moment. For users of Mac OS, they probably have to wait a little longer.

This thesis uses JDIC to provide the system tray icon (on the bottom right on Windows). The graphical user interface (GUI) will need the processor resources when it is active and for refreshing the GUI. When users minimize the application to the system tray, the system resources can be freed and available for other more intensive applications.

4.4.3 JCIFS

JCIFS [14] is an open source CIFS client library written completely in Java. With JCIFS library, it is possible to develop Java applications that can communicate with CIFS servers. Since Mobile Home Access is written in Java, JCIFS enables the web services to communicate with CIFS servers on the local area network.

4.4.4 APIs for Web Services

Below is a list of Java APIs relavant to XML Web Services and a short overview of the main purpose for each API or specification.

J2ME Web Service Specification
J2ME Web Service Specification, developed as JSR 172 [15], provides two optional packages to access remote SOAP/XML based web services and parse of XML data for mobile devices. This specification specifies subset of Java APIs and technologies of relevance to XML Web Services, such as WSDL, SOAP, JAXP and JAX-RPC, to the J2ME profiles.

Java APIs for XML Processing (JAXP)
Java APIs for XML Processing [16] provides XML parsing functionality for Java applications independent of XML processing implementation. JAXP allows us to choose parser standards (such as Simple API for XML Parsing - SAX and Document Object Model - DOM API) or XML-compliant parser. The main APIs are defined in *javax.xml.parser* package.

Java APIs for XML-based RPC
The Java APIs for XML-based RPC (JAX-RPC) [17, 18] provides a simple way to create remote procedure call-based web services without requiring the developer to be aware of the

way the SOAP messages encode and decode the procedure call requests and responses. It is developed as a standard specification under the Java Community Process, JSR-101.

SOAP with Attachments APIs for Java
SOAP with Attachments APIs for Java (SAAJ) [19] provides a convient way to construct SOAP messages, helping developers to avoid create the XML directly. SAAJ originated as part of the Java APIs for XML Messaging (JAXM) under JSR-67 [20], but has been splitted during the maintenance cycle for JSR-67 to make the low-level SOAP message creation features independent of JAXM.

Java APIs for WSDL
Java APIs for Web Services Description Language Specification [21] is a standard set of APIs that allows the creation, representation and manipulation of WSDL documents. WSDL4J is an example of reference implementation of the specification from IBM.

Chapter 5 Networking

Nowadays, computer networks facilitate communication between two or more computers and the distance between network nodes may vary from a few centimeters (e.g. via Bluetooth) to world wide (e.g. via the Internet). Many families simply use networking as a way to share an Internet connection. Unfortunately, few have discovered benefits/advantages of networking:

- sharing printers, scanners and other peripherals
- sharing documents and files
- sharing services

This chapter studies the home networking, special for Windows networks terminology. The purpose of networking technologies, firewall and NAT, are described in the context of Mobile Home Access.

5.1 Microsoft Network

Since most users have the Windows operating system, we will concentrate on Windows network's terminology. A windows network consists of PCs with Windows operating system (such as Windows 2000 and XP) installed.

Typical on large (e.g. enterprise) networks, there are often a couple of computers running Unix/Linux operating system. These computers can participate or even act as server for Windows networked file system by running Samba, an implementation of Microsoft CIFS protocol for Unix.

5.1.1 Windows Domain

Windows domain is a client-server network architecture. The server (domain controller) maintains information about users, computers and shared resources in a hierarchical and object-based fashion using directory service, Active Directory (AD). AD structure is a framework of objects, which fall into three categories: resources, services, and accounts. AD stores these information (objects and their attributes) and settings in a central and organized database.

In later releases of Windows, all domain controllers are considered equal and have full access to the accounts databases. Prior to Windows 2000 and NT 4, a server can be one of these roles:

- *Primary Domain Controller (PDC)*
  PDC is the central domain controller which contains the master copy of the user accounts database. Only one domain controller can have this role.
- *Backup Domain Controller (BDC)*
  There may be one or more Backup Domain Controller in a domain. The BDC has a read-only copy of the user accounts database which is replicated from the PDC. If the PDC for some reasons fails, one of the BDCs will become the PDC. The Backup Domain Controllers provide fault tolerance and load balancing the workloads in an intensive network environment.

A Windows Domain network, with a central user-database, is an advantage on larger networks with many users. This is obvious because there is only one single user-database to maintain. There is also no need to synchronize the user-database, especially when periodical changes of passwords are required for security reasons.

Home users must run a Windows server operating system and set them up correctly. This may be a difficult task. Microsoft has another solution for home users that is more suitable for home networking, i.e. using Windows Workgroup.

5.1.2 Windows Workgroup

A Windows workgroup is a peer-to-peer network architecture, in which computers are considered equally and function as both client and server simultaneously. Windows workgroup consists of a collection or a group of computers where each computer has its own user-database, i.e. no central distributed database. Each computer is responsible to maintain its own resources, services and user accounts.

The Microsoft Windows family of operating systems supports assigning computers to named workgroups. When the Windows operating system is installed, it is automatically set on a default workgroup which is normally named *WORKGROUP* or *MSHOME*. This makes it easier for home users to set up.

Workgroup is recommended (by Microsoft) for home networks and small offices for (simple) sharing of files and resources. When the number of computers increases, it is too difficult to manage as users must keep the information synchronized between computers. In such cases, client-server (Windows domain) network architecture is recommended to eliminate this problem.

Note:
As there are only a few computers in home network, it is assumed that home networks are configured as Windows workgroup as it is simple to set up.

5.2 Firewall

A firewall is a software program, hardware device or combination of both that protects a network or computer from malicious users and applications. The firewalls will block all incoming traffic to services it considers insecure. In home network, firewalls are normally installed in the router and on personal computers. In Windows XP with Service Pack 2, Windows Firewall [22] is a built-in application.

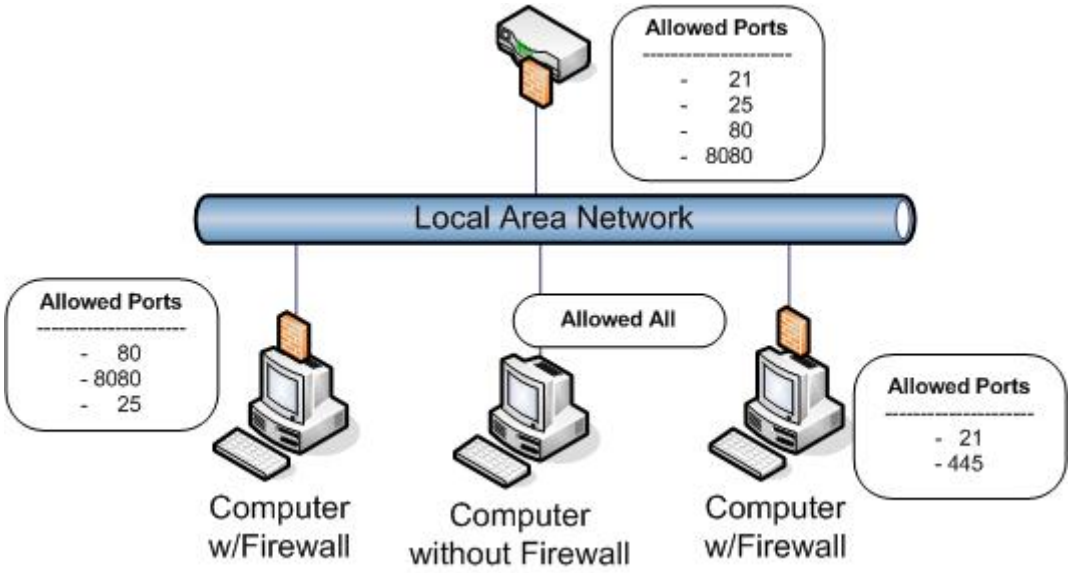

Figure 5.1 Firewalls in Home Networks

A typical firewall configuration, will block requests at well-known port numbers [23] (port number from 0 thtrough 1023), because services at these ports are used by system processes or programs executed by privileged users. To allow incoming connection to a service, ports for the specific program or service must be open.

In figure 5.1, the network firewall (firewall running on router or other network devive) filters traffic to/from the local area network. The network firewall can be compared as a front door to our home, where it controls the access to the internal services. The personal firewall (firewall running on a single computer) filters traffic to/from a computer. Most personal firewalls notify the users when an application wants to open a new port and give us the options (for example permit, block, and permit always) to control how these traffic should be handled. On computers without firewall, all traffic are allowed.

5.3 Network Address Translator

Network Address Translation (NAT) [24], also known as IP masquerading, is a technique where the source or destination IP address of data packets are changed as they pass through a router or firewall. NAT is designed to overcome the shortage of IPv4 addresses. Theoretically, there are 2^{32} possible IP addresses, but the actual number of addresses is smaller (~3.1 billion) because some addresses are aside for multicasting and reserved for testing or other special purposes. Behind a NAT device, hundreds of PCs and servers can masquerade as a single public IP address. This increases the number of devices that can access the Internet without running out of public IP addresses.

A home network is normally assigned one public IP address, and uses NAT to enable multiple computers to access Internet. Home PCs are assigned private IP addresses that cannot be routed over the Internet. An advantage to this is that hackers do not know the correct (private) IP address of the target computer, and therefore prevents from being directly attacked as packets sent to private IP addresses will never pass over the Internet.

5.3.1 Port forwarding

Using NAT break the end-to-end flexibility by means that computers outside the local area network (i.e. the Internet) cannot access internal computers directly. Traffic with private IP

addresses will never be routed to the Internet, and will be dropped at the router. The most common solution is using the Network Address Port Translator (NAPT) [25].

NAPT is an extension to NAT in that it uses TCP/UDP ports in addition to IP addresses to map many private network addresses to a single public IP address. NAPT makes internal services at home network available for others outside the home network. The mapping from external ports number to internal service is a one-to-one relationship. This means that only one service can be exposed at a specific port.

Figure 5.2 is an example that enables three services, FTP, SSH, and web server on different computers in the local area network. Remote users access these services by sending requests to the public IP address (in this example 129.241.12.22) and the service port.

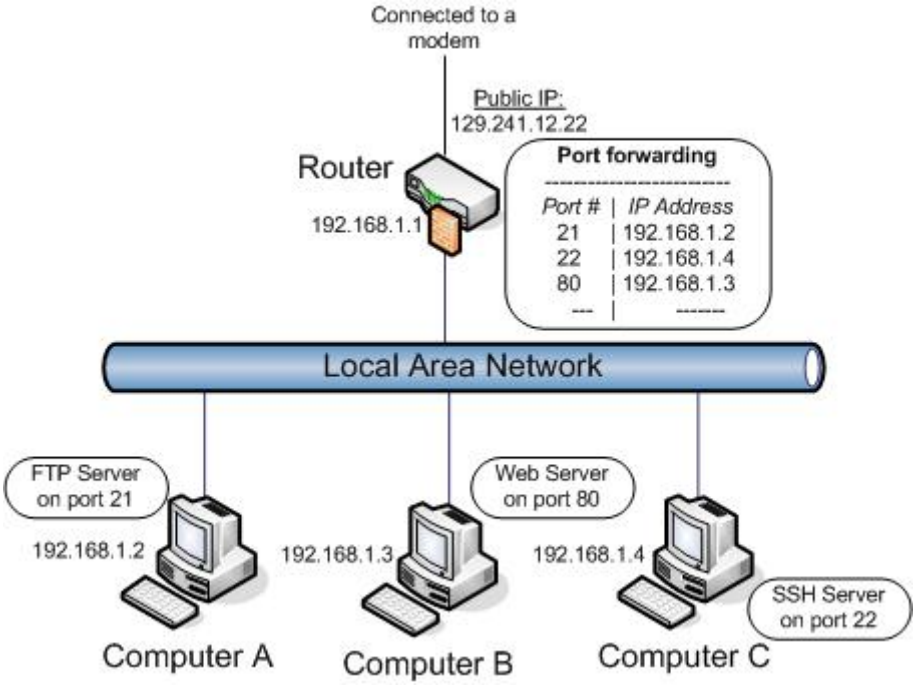

Figure 5.2 NAT Port Forwarding

Chapter 6 CIFS

Common Internet File System (CIFS) [26, 27] is a networked file system for sharing files, printers and other network resources over local area network. CIFS is Microsoft's enhanced version (some will refer it as a dialect) of Server Message Block (SMB) protocol developed by IBM.

Network File System (NFS) [28], another alternative to CIFS, is designed to work for all platforms, but it is more associated with Unix-based systems. The dominance of Windows operating systems on the personal computer market is the main reason of why CIFS has quickly become popular even though the lack of specification.

6.1 NetBIOS

CIFS services rely on the NetBIOS (Network Basic Input Output System), that allows applications on different computers to communicate within a local area network (LAN). NetBIOS was developed by IBM and Sytek Inc. and includes both a set of services and an API (Application Programmer Interface) to those services. Microsoft could not abandon NetBIOS completely as many of their software relied on it. However, Microsoft introduced NetBIOS over TCP/IP (NBT) [29, 30] to allow NetBIOS to run over IP network.

NetBIOS consists of three services:

- *Name Service*
- *Datagram Service*
- *Session Service*

NetBIOS Name Service

The NetBIOS Name Service (NBNS) is a service for name registration and resolution. NetBIOS names are 16 bytes long (if a name is shorter than 15 characters, it is padded with spaces), and the 16th byte is used to indicate a particular service such as CIFS. NetBIOS names may contain non-alphanumeric characters, although some implementations do not support this. The structure of NetBIOS namespace is flat and a NetBIOS name must be unique in local area networks.

Functions or service primitives offered by the name service are:

- *Add Name* – registers a NetBIOS name
- *Add Group Name* – registers a NetBIOS group name
- *Delete Name* – unregister a NetBIOS name or group name
- *Find Name* – looks up a NetBIOS name on the network

NetBIOS Session Service

NetBIOS Session Service is a full-duplex, sequenced, and reliable transport service that lets two computers establish a connection. The session service has three states/phases:

1. *Session establishment*
   Establish the NetBIOS session with remote server
2. *Data exchange*
   NetBIOS messages can now exchange over the session

3. *Session close*
   A session is closed when a party closes the session or one of the parties has gone down.

Functions or service primitives offered by session service are:

- *Call* – opens a session to a remote NetBIOS name
- *Listen* – listen for attempts to open a session to a NetBIOS name
- *Hang Up* – close a session
- *Send* – sends a packet to the computer on the other end of a session
- *Send No Ack* – similar to Send, but doesn't require an acknowledgment
- *Receive* – wait for a packet to arrive from a Send on the other end of a session

NetBIOS Datagram Service

The NetBIOS Datagram Service is connectionless, unreliable, non-sequenced transport service. The Datagram Service is used to broadcast computer's NetBIOS names and its services to the local area network.

Functions or service primitives offered by datagram service are:

- *Send Datagram* – send a datagram to a remote NetBIOS name
- *Send Broadcast Datagram* – send a datagram to all NetBIOS names on the network
- *Receive Datagram* – wait for a packet to arrive from a Send Datagram operation
- *Receive Broadcast Datagram* – wait for a packet to arrive from a Send Broadcast Datagram operation

6.1.1 NetBIOS over TCP/IP

NetBIOS was designed to operate in LAN and not meant to be used in WAN. The flat namespace in NetBIOS makes it non-routeable over WAN. In addition, computers identified by names (15 characters) is effective on a network consisting of few computers. Nowadays networks tend to consist of many computers which communicate across WAN. Thus there is not possible to avoid naming conflicts and solve the routing between subnets using NetBIOS.

Fortunately, NetBIOS is transport independent and could bind to any transport protocols (see NetBIOS protocol stack in the table below). The most interesting is NetBIOS over TCP/IP because it employs a hierarchal namespace using IP addresses to route packets across WAN.

<table>
<tr><th>Layer</th><th>OSI</th><th colspan="3"></th></tr>
<tr><td>7</td><td>Application</td><td colspan="3" rowspan="2">SMB/CIFS</td></tr>
<tr><td>6</td><td>Presentation</td></tr>
<tr><td>5</td><td>Session</td><td>NetBIOS</td><td colspan="2">Named Pipes</td></tr>
<tr><td>4</td><td>Transport</td><td rowspan="2">IPX/SPX</td><td>TCP</td><td>UDP</td></tr>
<tr><td>3</td><td>Network</td><td colspan="2">IP</td></tr>
<tr><td>2</td><td>Data Link</td><td colspan="3">Ethernet, 802.x, etc</td></tr>
<tr><td>1</td><td>Physical</td><td colspan="3"></td></tr>
</table>

Table 6.1 NetBIOS/CIFS Protocol stack

NBT encapsulates NetBIOS messages within TCP and UDP packets. The TCP/IP protocols functions as a carrier for local NetBIOS that allows NetBIOS to be sent between networks (LANs), WAN. NetBIOS services over TCP/IP communicate via TCP and UDP ports, see the figure and table below.

Service	Keyword	Port	TCP/UDP
Name Service	netbios-ns	137	UDP (TCP[6])
Datagram Service	netbios-ssn	138	UDP
Session Service	netbios-dgm	139	TCP

Table 6.2 NetBIOS Services

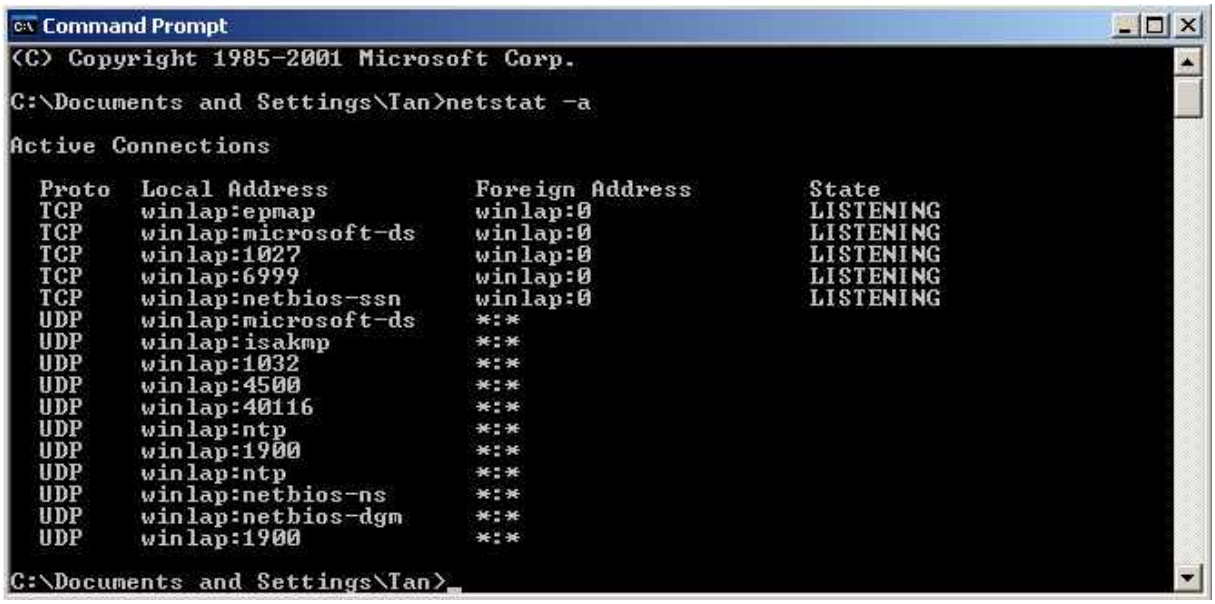

Figure 6.1 NetBIOS ports on Windows

NetBIOS Name Service and Datagram Service runs in UDP, a connectionless protocol which means that these services do not need an established connection. Session Service which runs on TCP port requires a TCP connection to be established before messages could be exchanged. The Session Service provides a one-to-one mapping of NetBIOS sessions to TCP sessions. Through the TCP connection, a client initiates a NetBIOS session by sedning an NBT SESSION REQUEST message and recieves an NBT POSITIVE SESSION RESPONSE message indicates that the NetBIOS session is established.

6.2 Name Resolution

Before the NetBIOS session can be established, the client (calling node) must discover the IP address of the server (called node). Name resolutions of NetBIOS names to IP addresses are done using the NetBIOS Name Service. Name resolution on a home network is accomplished by broadcasting (similar to ARP protocol) which could be compared to calling out a person's name in a crowd. When a computer wants to look up, say a computer named *Per*, it broadcasts a message to all computers on the local area network saying "*Per, where are you? Answer me with your IP address*". Computer *Per* then answer with its IP address, other computers simply ignore this message.

Broadcasting causes a lot of traffic on the network, a better approach is using a Windows Internet Naming Service (NBNS server) where computers register their names and IP address. WINS employs a distributed client-server system to maintain the mapping of computer names

[6] NetBIOS Name service runs normally on UDP port, but it is possible to run on TCP.

to addresses. Windows clients can be configured to use primary and secondary WINS servers that dynamically update name/address pairings as computers join and leave the network.

Note:
On a TCP/IP network, if the router is not configured to forward the name registration and name query broadcasts, computers on different subnets will not be able to see each other.

Because there are several approaches to resolve names, Microsoft has defined the order of resolution:

- Check if the name is the local machine name
- Check the NetBIOS name cache
- Query the WINS server (if it exists)
- Broadcast the local network
- Check LMHOSTS file (if configured)
- Check HOSTS file and then DNS server if configured

SMB names are directly mapped to NetBIOS names. The 16th byte is important for identifying various services and is known as the *suffix*.

NetBIOS/SMB Name	Purpose
<01><02>__MSBROWSE__<02><01>	Registred by the Master Browser
Computername [0x00]	Workstation service
Computername [0x20]	Server service
DOMAIN(00)	Register computer in domain
DOMAIN(1B)	Registered by PDC
DOMAIN(1C)	Domain controller
DOMAIN(1D)	Registred by Master Browsers
DOMAIN(1E)	Registered by all Browser servers and Potential Browser Servers in a domain/workgroup

Table 6.3 SMB Names

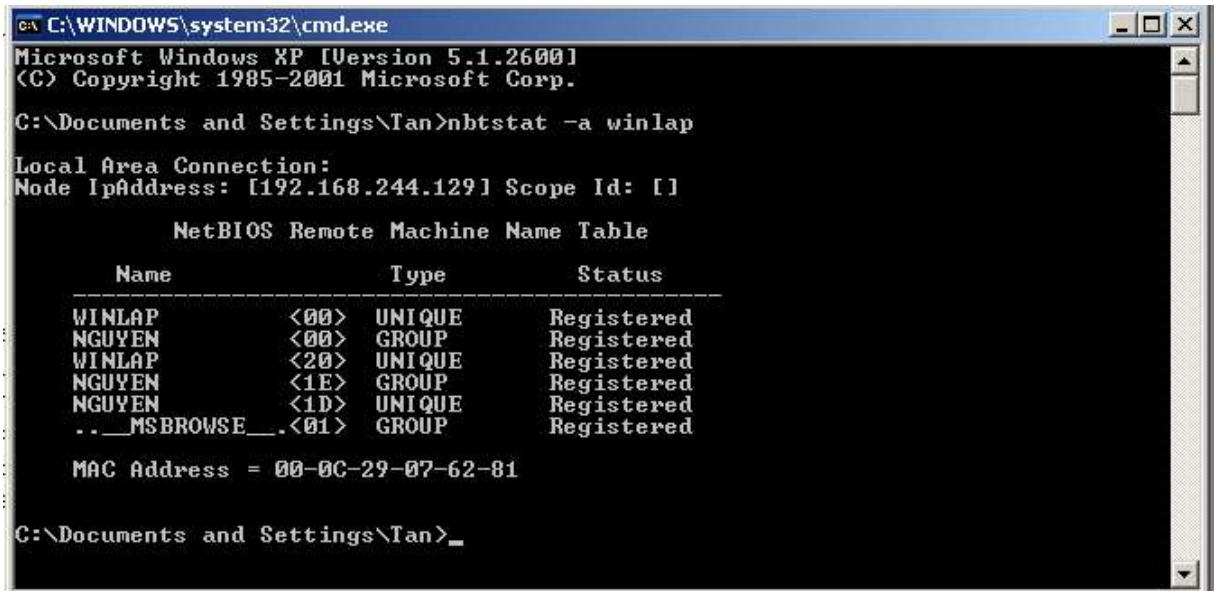

Figure 6.2 NetBIOS Names example

6.2.1 Universal Naming Convention Format

The Universal Naming Convention (UNC) [31] format provides a naming convention for identifying network resources. UNC names consist of three parts, a server name, a share name, and an optional file path, that are combined using backslashes as follows *\\server\share\file_path*.

The server portion of a UNC path refers to names maintained by a network naming service such as DNS or WINS. Share names can be defined by a system administrator or, in some cases, exist automatically within the local operating system. Predefined share names in Windows generally end with a *$*, but this convention is not required for any new shares an administrator defines.

Share name	Purpose
ADMIN$	Remote admin
IPC$	Remote Inter-Process Communication (IPC) is used for data sharing between applications and computers.
C$	Default share
SharedDocs	Shared documents between users
Print$	Printer drivers

Table 6.4 Predefined Share Names

6.2.1.1 SMB URI

The SMB format looks similar as a URL-address as you can see on the address bar on a web browser (for example http://www.ntnu.no). SMB URI Scheme IETF draft [32] proposes two different schemes: cifs and smb. The latter appears to be most popular (in Samba in Unix-based environment and JCIFS) and is the preferred form.

Syntax and semantics:

- **smb://**
  interpreted as a request for a list of Workgroups available on the local LAN
- **smb://:3242/** (UNC: not spuported)
  Request browse list (workgroups) using destination port 3242
- **smb://server** (UNC: *\\server*)
  This form could specify a CIFS server or Workgroup. If it is a CIFS server, then this is a request of a list of shares on the server. If the name resolves to a workgroup name, then this is a request of a list of CIFS servers within the workgroup.
- **smb://server/path/to/share/file.zip** (UNC: \\server\path\to\share\file.zip)
  Indicates file on server
- **smb://username@server/path/to/share/file.zip** (UNC: not supported)
  Indicates file on server and connect using username

6.3 Security

Authentication in CIFS is categorized as follows:

- **None**
  No authentication, everybody can access the resource

- **Share-based**
  A share-based resource uses a password for authentication. Everybody that knows the password have access to the resource.
- **User-based**
  A user-based resource requires the client to provide a username and corresponding password to gain access. User-based servers may use the account name to check access control lists on individual files, or may have one access control list that applies to all files in the directory.
- **Authentication Server-based**
  (NT Domain and Kerberos)

6.4 Service Announcements/Browse Service

My Network Places (also known as Network Neighborhood in older version of Windows) is a network browser which provides browsing of computers and resources on a local area network. My Network Places is based on the CIFS browse service or service announcements [33].

On a Windows network, there must be one computer, for every workgroup/domain, acting as a *Master Browser*, which stores all share resources on every computer on the network. The browser election is the first step, when there is no Master Browser present.

The election process will choose the strongest system/computer to become the Master Browser when the network is a mixture of different Windows versions. For example, Windows XP is stronger than Windows 98. A computer can also be configured to automatically become the Master Browser and therefore avoid the election process. Generally, a server is always stronger than a workstation.

Windows Versions	Priority
Windows 3.11	Low
Windows 95	Higher
Windows 98	Higher
Windows NT	Higher
Windows 2000	Higher
Windows XP	Higher
Windows NT Server	Higher
Windows 2000 Server	Higher
Windows 2003 Server	Highest

Table 6.5 Master Browser election criterias

Note:
In Windows Domain, a computer that acts as the Primary Domain Controller is “always” (has higher priority in browse elections) chosen as the Domain Master Browser.

When a Master Browser is present or elected, every computer on the same workgroup will announce their list of share resources to it. Opening and browsing Network Neighborhood/My Network Places will send a request to the Master Browser, which responds with the list of available resources or responds with an error message.

For every 32th computers on the network, there is one Backup Browser which can become Master Browser if necessary. Other computers will become either a Non-Browser server or Potential Browser (which can become Backup Browser if necessary). On Windows Domain, there is one Domain Master Browser which distributes the browse list between Master Browsers across subnets. Thus, a CIFS server can have one of the browse roles in table 6.6.

Role name	Description
Non-Browser	Servers which do not maintain browse list
Potential Browser	Servers which can become a Browser server if needed
Backup Browser	Maintains a copy of a browse list from Master Browser
Master Browser	Maintains the browse list and send it to Backup Browser in addition to promote Potential Browser to Backup Browser if needed.
Preferred Master Browser	Same as Backup Browser, but has higher precedence in browser election. This role is only available if the Registry is set to support Preferred Master Browser.
Domain Master Browser	Receives Master Browser's (from each subnet) announcements to be able to browse across WAN connections. The PDC has special biased in browser elections to become Domain Master Browser.

Table 6.6 Browser Roles

If the server or domain names in the browse list are inactive (in three Announce periods), the Browser sends a message using Mailslot Transactions to remove the names from the browse lists. MAILSLOT\LANMAN is used for all systems including LAN Manager and Windows for Workgroups while MAILSLOT\BROWSE is used for messages intended for Windows NT server.

Regular users often experience strange behavior in Windows Browse service. Typical in network environment with many different versions of Windows and/or the network settings are not configured correctly. Knowing the principles of how Windows networking works help to understand the strange behaviors (more details can be found in Appendix B).

6.5 CIFS Protocol Operation

When a CIFS client has successfully established a NetBIOS session (using a TCP connection), it exchanges SMB (Server Message Block) messages with the server to access the resources. The sequence diagram below shows an example of SMB message exchange. The first message a client will send to the server is an *SMB_COM_NEGOTIATE* to negotiate the SMB/CIFS dialects to be used in subsequent requests. Then the client setup the session with an *SMB_COM_SESSION_SETUP_AND_X* message.

After a session is successfully setup, the client can connect to a disk share with *SMB_COM_TREE_CONNECT* or send other SMB messages. Note that performance can be increased or at least reduce the round trips by combining SMB messages into one through the batching mechanism (called *AndX*).

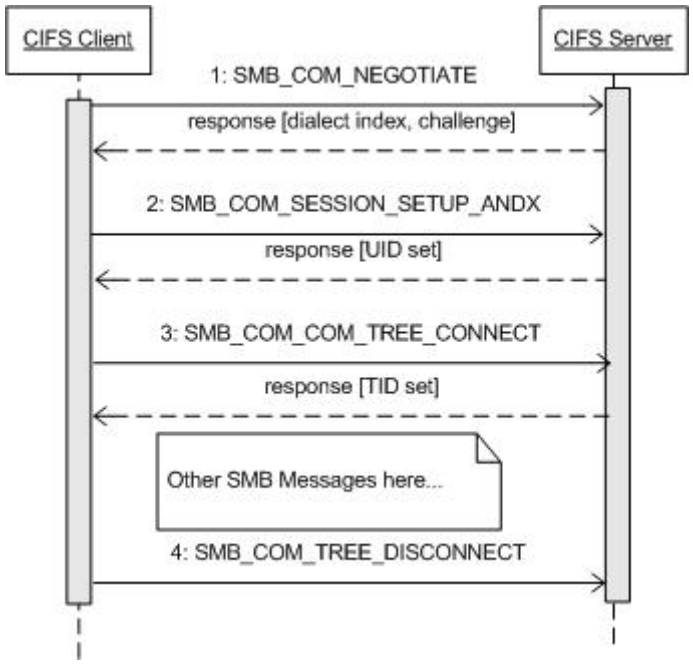

Figure 6.3 CIFS Message Exchange

6.5.1 Protocol Negotiation

The client sends a request (*SMB_COM_NEGOTIATE*) with a list of dialects it understands. The server then responds with the index of the dialect that it wants to use, or 0xFFFF if none of the dialects were acceptable. The PC NETWORK PROGRAM dialect is the core protocol and must be understood by all CIFS implementation. Other dialects are included in the table below.

Dialect Name	Reference
NT LM 0.12	
LANMAN2.1	
LM1.2X002	Extended 2.0 protocol
Windows for Workgroups 3.1a	
LANMAN1.0	Extended 1.0 protocol, first version of full LANMAN 1.0 protocol
MICROSOFT NETWORKS 3.0	Extended 1.0 protocol
MICROSOFT NETWORKS 1.03	Core plus dialect
PC NETWORK PROGRAM 1.0	Core protocol

Table 6.7 SMB Dialects

The response from the server also includes authentication information (a challenge), which the client must respond to verify the client's identity. The clients can send the response to the challenge in *SMB_COM_TREE_CONNECT*, *SMB_TREE_CONNECT_ANDX* and one or more of the *SMB_COM_SESSION_SETUP_ANDX*.

6.5.2 Session Setup

The client initiates a session in a *SMB_COM_SESSION_SETUP_ANDX* message. This message includes usernames and credentials to the server for verification. Upon successful authentication/verification, the server's response message has the UID field set in the SMB header. The UID field is included in subsequent SMB messages on behalf of the user.

6.5.3 Connect to a Resource

The user then accesses the resource using *SMB_COM_TREE_CONNECT* which includes the name of the disk share available on the server. On *winlap*, the author's test computer, there are two disk shares available depicted in the figure below.

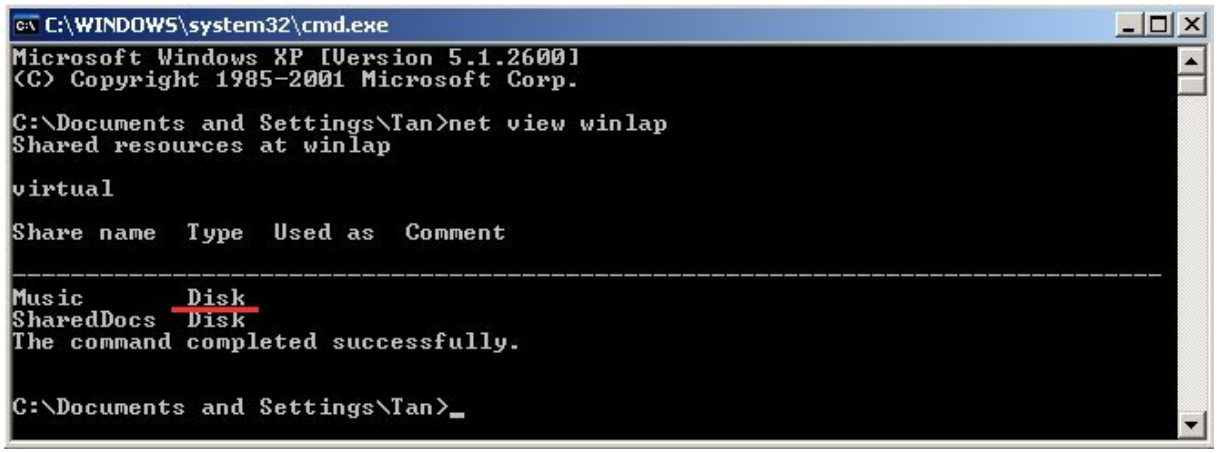

Figure 6.4 Disk shares on *winlap*

Server response has TID field set in the SMB header, which is used for subsequent SMB messages referring to this resource. The user can execute other operation on the connected resource, such as read, open, delete a file. Finally, user can close the resource using an *SMB_COM_TREE_DISCONNECT* message to inform the server that the client wants to disconnect from the resource represented by TID value.

6.6 Samba

Samba [34] is an open source software suite that provides seamless file and printer service. The primary goal of Samba is to remove the interoperability problem to enable other systems (Unix-based systems) to interact with Microsoft CIFS clients or servers. In fact, Samba is reverse engineered (initiated by Andrew Tridgell) from the SMB/CIFS protocol. Samba has grown in popularity because of its flexibility and the freedom in terms of setup and configurations.

Samba consists of two key programs, *smbd* and *nmbd*, that implement the services in CIFS protocol.

Program	Service(s) implemented
smbd	Core SMB/CIFS services, i.e. file and print services, and authentication and authorization
nmbd	Name resolution and browsing

Table 6.8 Samba core programs

Samba includes a lot of utilities.

Utilities	Functionality
smbclient	A simple SMB/CIFS client
nmblookup	A NetBIOS name service client for name resolution and browsing
swat	The Samba Web Administration Tool allows users to configure Samba remotely from the web server.

Table 6.9 Samba utilities

As Microsoft completely determines how CIFS will work, the Samba Team continuously works to support new functionalities and maintains compatibility in Samba. Thus, Samba can be viewed as a clone of the CIFS protocol for Unix-based computers.

Additional features of Samba:

- Can act as a PDC
- Can act as a WINS server
- Active Directory integration, i.e. acts as an Active Directory domain member server
- Can join a Windows NT/2000/2003 PDC
- Supports the use of multiple account databases, which could be distributed and replicated
- Supports authentication for Windows domain logins and provide flexible authentication mechanisms

Chapter 7 XML Web Services

Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and frameworks. Web Service uses a collection of widely-used protocols and standards, which are defined or specified by extensible markup language (XML) [35]. In addition to the benefits XML provides; XML-based communication makes Web Services independent of any operating system or programming language as well as being easily extended and automated.

The Web Service Architecture [36] is in fact an implementation of the Service-Oriented Architecture (SOA) [37], which is essentially a collection of loosely coupled services and the communication and interaction between the consumer and the service provider. Web Service implements the communication using XML vocabulary.

The interaction between different parties is shown in the figure below. The figure is self-explanatory, where in step 1 the service provider registers the services to a central directory. Consumers then query the directory for available services and get a description of the service in form of WSDL. Using the description of the service, service consumer construct requests and send it to the service providers.

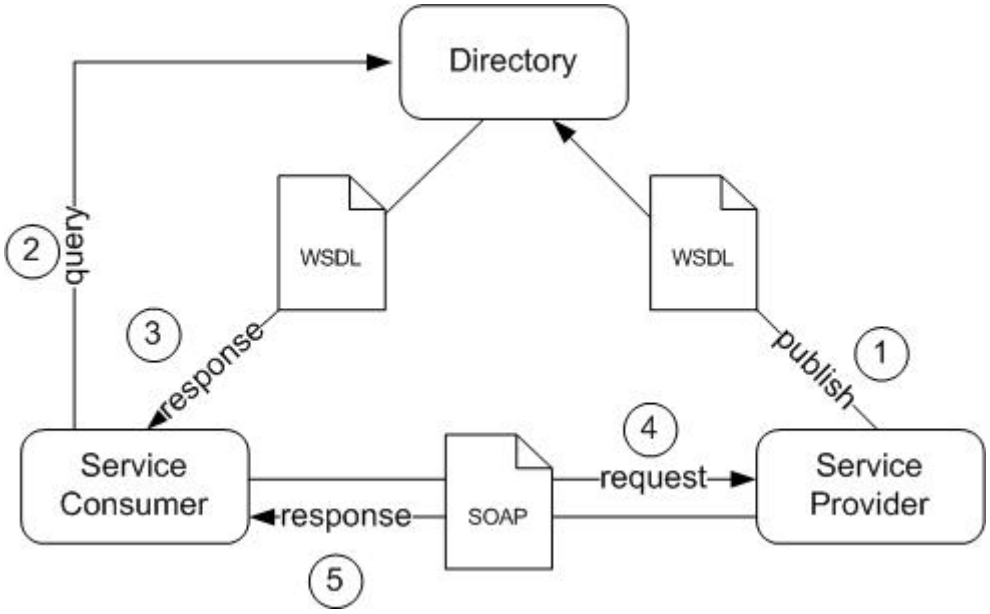

Figure 7.1 Web Service Architecture

In the Web Service Architecture, the following open standards are typically used:

- *Extensible Markup Language (XML)* to tag the data. The data must conform to well-formed XML syntax.
- *Hypertext Transport Protocol (HTTP)* is used as carrier of SOAP messages.
- *Simple Object Access Protocol (SOAP)*, a light-weight protocol used to transfer the data
- *Web Service Description Language (WSDL*), is used for describing the services
- *Universal Description, Discovery and Integration (UDDI),* is used for listing and discover web services

7.1 SOAP

SOAP (Simple Object Access Protocol) [38] is a protocol for exchanging structured information or messages between software applications. Compared to other frameworks, such as CORBA, DCOM and Java RMI, which provide similar functionality, SOAP is an XML-based standard that describes the contents of a message and how to process it. It is therefore independent of transport protocols and platform.

SOAP represents a cornerstone of the Web Service Architecture. All the messages in figure 7.1 are carried by SOAP. Using SOAP enables diverse applications to easily exchange services and data. SOAP also includes rules for how SOAP messages can represent RPC calls and responses. Similarly, XML-RPC provides a simple XML-based mechanism for making method or function calls across a network, but SOAP is more extensible and flexible.

7.1.1 SOAP message

A SOAP message contains the following elements:

- A mandatory **Envelope** element that identifies the XML document as a SOAP message
- An optional **Header** element provides a mechanism for extending a SOAP message such as information for routing, security and transactions
- A mandatory **Body** element that contains the main information or commonly known as payload. The SOAP Body element can have one particular child, the SOAP **Fault**, which is used for reporting errors

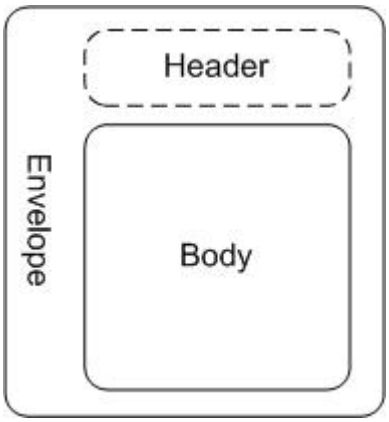

Figure 7.2 SOAP Message

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instane">
   <soap:Body>
        <m:GetPrice xmlns:m="SOM-URI">
           <m:ItemNo>435323</m:ItemNo>
        </m:GetPrice>
   </soap:Body>
</soap:Envelope>
```

Table 7.1 Example of a SOAP message

7.1.1.1 *Envelope* Element

The *Envelope* is the root element in a SOAP message. It has namespace attribute to differentiate version of SOAP as well as other namespaces.

Version	Namespace URI

SOAP 1.1	http://schemas.xmlsoap.org/soap/envelope/
SOAP 1.2	http://www.w3.org/2001/09/soap-envelope

Table 7.2 SOAP versions and namespaces

7.1.1.2 *Header* Element

The *Header* element is optional and offers a flexible framework for specifying additional application-level functionalities and requirements. For example, the *Header* element can be used for authentication (digital signatures, etc) and routing.

The protocol specifies two attributes for *Header* element:

- *actor*
  Specifies the recipient of the SOAP header.
- *mustUnderstand*
  By setting this attribute to *true*, the recipient must understand and process the *Header* attribute according to its defined semantics. A SOAP Fault message (faultcode: SOAP-ENV:MustUnderstand) is returned if the recipient is unable to process the *Header* element.

7.1.1.3 *Body* Element

The *Body* element encapsulates the main payload of the SOAP message. In the event of an error, the *Body* element will include a *Fault* element. *Fault* element has the following sub-elements:

- *faultCode*
  indicates the class of error:
  - `SOAP-ENV:VersionMismatch`
    Invalid namespace for SOAP Envelope element is found
  - `SOAP-ENV:MustUnderstand`
    Could not understand child elements of the Header element with mustUnderstand attribute set to true
  - `SOAP-ENV:Client`
    Indicates that the message is incorrectly formed or contained incorrect information
  - `SOAP-ENV:Server`
    Indicates internal server errors while processing the message
- *faultString*
  Explanation of the error.
- *faultActor*
  indicates in which node caused the error. This is useful when the SOAP message travels through several nodes.
- *detail*
  contains application-specific error message related to Body element

7.1.2 SOAP encoding

The message format must be defined so that a sender could construct SOAP messages and receiver could know how to process them. The format of a message includes a set of elements, and corresponding name, type, and attributes for each element. A sender constructs a SOAP message which must conform to the defined format. The definition of message format is defined using XML Schema [39] vocabulary and may be included in the WSDL document. XML Schema can define simple data type (scalar types such as integer, boolean,

String, etc) and complex data type (compound data types such as array of String, array of bytes, etc).

When the application code already exists, the development of WSDL documents is not the preferred starting point. In that case, the message format is generated using SOAP encoding. The SOAP encoding includes a set of rules to map from programmatic data types to XML. These rules specify how data type would be serialized (by the sender) and deserialized (by the receiver). Soap encoding is specified using the *encodingStyle* attribute, which could appear in Header block, child element of the Body elements and child element of Detail element. The value of the *encodingStyle* attribute indicates that SOAP encoding is used to serialize the message and receiver simply follows those rules to deserialize it.

The standard value/URI for *encodingStyle* is http://www.w3.org/2001/12/soap-encoding (for SOAP 1.1), which maps to XML Schema data types. Other encoding styles can be used by specifying the correct URI value.

7.1.3 Transport binding

SOAP Binding Framework [40] offers transport binding to a variety of transport protocols (such as HTTP, SMTP, FTP, JMS or RMI/IIOP) for exchanging messages. Remote procedure call request and response can also be packed into SOAP messages and interact between clients and server. This facilitates a platform-independent and distributed system.

SOAP is commonly bound to HTTP transport protocol. A SOAP message could use a HTTP POST or HTTP GET Web method. SOAP request and response messages are mapped directly to HTTP requests and responses, respectively. SOAP imposes the use of two HTTP headers, *Content-Type* and *SOAPAction*. The Content-Type header specifies that the payload is a SOAP message. The media type must be *application/soap+xml*. The SOAPAction header is used to identify the intent of the call, but most implementations ignore it.

```
POST /axis/services/EchoService HTTP/1.1
HOST www.example.com
Content-Type: application/soap+xml; charset=UTF-8
SOAPAction: ""
Content-Length: 567

(SOAP message here)
```

Table 7.3 Example of TCP header for SOAP (request) message

HTTP response codes are used to indicate the status of requests. A 200 code indicates that the request is accepted and processed, whereas a 500 code, Internal Server Error, indicates that the server could not process the request (the response includes a SOAP Fault).

7.1.4 SOAP with Attachment

A SOAP message may need to transmit binary data, for example an image or document, along with SOAP message. The simplest method is using XML Schema type *base64binary* to encode data inside the XML.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instane">
   <soap:Body>
        ...(other contents)
        <image imageType="jpg" xsi:type="base64binary">
```

```
            4f32f32...
         </image>
   </soap:Body>
</soap:Envelope>
```

Table 7.4 SOAP with binary data

This technique is inefficient in terms of bandwidth and processing time to encode and decode to and from base64 format. The SOAP with Attachment (SwA) Recommendation [41] describes how to use standard Multipurpose Internet Mail Extension (MIME) [42] mechanism (which has been used for a long time for attachments in emails) to encode 8-bit binary. The basic idea is to place a reference to data in the SOAP message and using MIME Multipart/Related [43] media type for encapsulation. This is more efficient than encoding/decoding data directly.

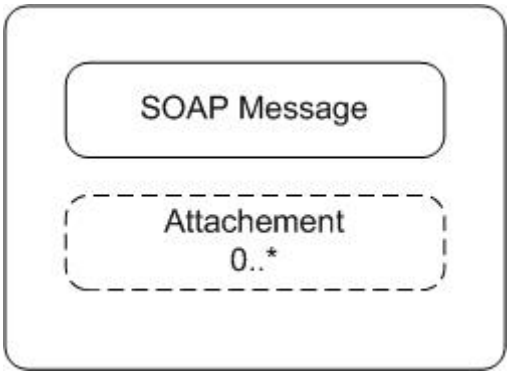

Figure 7.3 SOAP package

A SOAP message package (i.e. SOAP message with attachments) is constructed using the Multipart/Related MIME media type and the primary SOAP message must be the root body part. The *Content-Location* or *Content-ID* header in MIME is used to reference the MIME parts.

- *Content-ID* MIME header structure in the root part of the SOAP message
- *Content-Location* MIME header structure

For example to transmit an image similar as above, the SOAP message would be as follow:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary;
 type=application/soap+xml;start="<361AE9785992CDBC79899BAEC7409715>";

--MIME_boundary
Content-Type: application/soap+xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <361AE9785992CDBC79899BAEC7409715>

<soap:Envelope
 xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
 <soap:Body>
     ...(other contents)
     <image href="cid:785361A9E98999EC742CDBC79BA09715"/>
 </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/jpeg
```

```
Content-Transfer-Encoding: binary
Content-ID: <785361A9E98999EC742CDBC79BA09715>

...binary JPG image...

--MIME_boundary--
```

Table 7.5 SOAP with attachments

Direct Internet Message Encapsulation (DIME) [44, 45], another technology from Microsoft and IBM, uses a similar technique except that on-the-wire encoding is smaller and more efficient than MIME. DIME increases the performance in terms of parsing (small and fixed set of headers), encoding (data encoding not required), and memory allocation (because data length could be easily calculated and thus allocate memory). DIME is also designed for simplicity whereas MIME is designed for flexibility. This makes developing of tools for DIME easier.

Note:
Though the performance and simplicity of DIME may be tempting, the specified system chooses the flexibility in using MIME technique. This is regarding to future development of (new) services for the specified system. It is desired that the Mobile Home Access should be flexible and easily extensible.

7.2 WSDL

While SOAP is used to exchange messages in the Web Service Architecture, Web Service Description Language [46] is a language for describing services. A service is described as a set of communication endpoints, or ports, capable of exchanging messages. The operations and messages of an endpoint are described abstractly (i.e. independent of concrete implementations), and then bound to a standard network protocol (such as HTTP and SMTP) and message format to define an endpoint. WSDL is extensible because the formal description of endpoints and their messages are regardless of what message formats or network protocols are used to communicate.

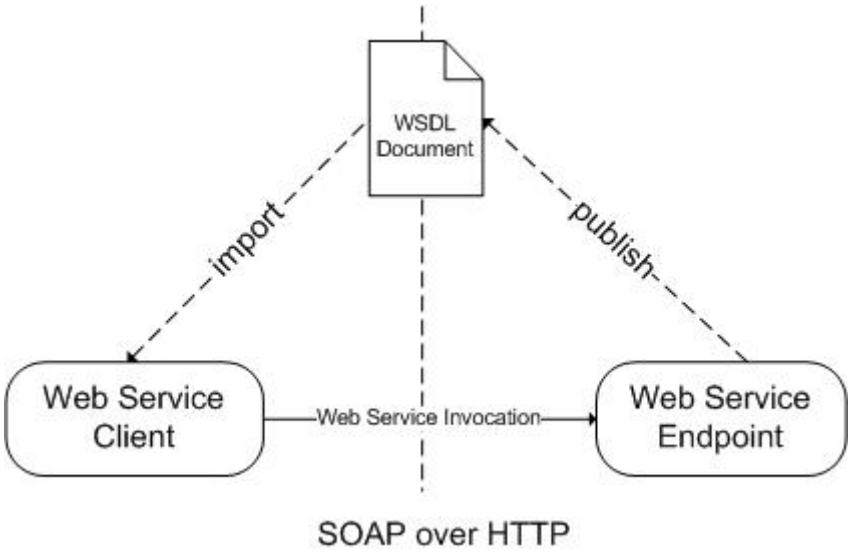

Figure 7.4 Web services

A web service endpoint publishes WSDL document to clients which uses definitions in WSDL to construct messages (i.e. SOAP message). Thus, a WSDL document can be viewed as a contract from a service provider where it says what services are available and how to request/use these services.

WSDL Elements

A WSDL document definition of network services consists of the following six elements and other utility elements such as *documentation* and *import*). XML Schema to WSDL documents is located at: http://schemas.xmlsoap.org/wsdl/.

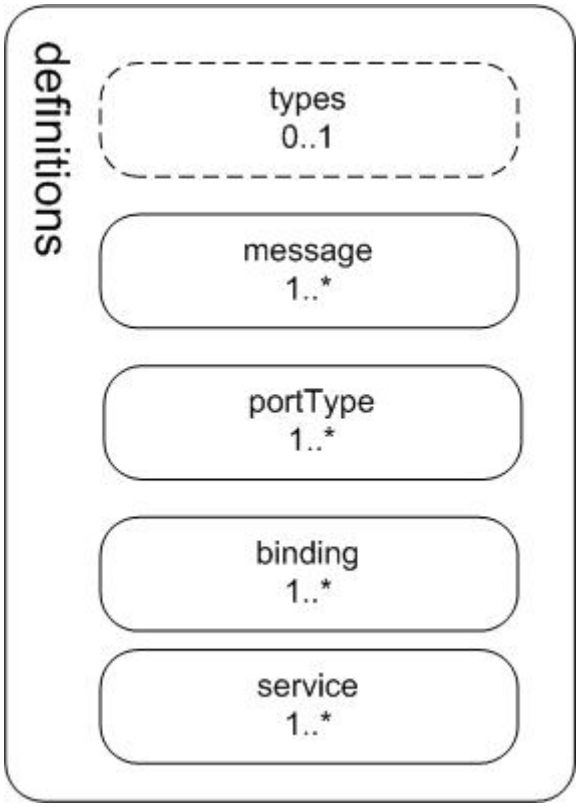

Figure 7.5 WSDL document structure

7.2.1.1 definitions Element

The *definitions* element is the root element of a WSDL document. This element is used to specify the namespaces in the WSDL document.

Prefix	Namespace URI	Definition
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL namespace for WSDL framework
soap	http://schemas.xmlsoap.org/wsdl/soap/	WSDL namespace for WSDL SOAP binding
http	http://schemas.xmlsoap.org/wsdl/http/	WSDL namespace for HTTP transport binding
mime	http://schemas.xmlsoap.org/wsdl/mime/	
soapenc	http://schemas.xmlsoap.org/soap/encoding/	Encoding namespace as defined by SOAP 1.1
soapenv	http://schemas.xmlsoap.org/soap/envelope/	Envelope namespace as defined by SOAP 1.1

xsi	http://www.w3.org/2000/10/XMLSchema_instance	Instance namespace as defined by XSD
xsd	http://www.w3.org/2000/10/XMLSchema	Schema namespace as defined by XSD
tns	(custom)	This namespace (tns) is used to refer to the current document

Table 7.6 WSDL Namespace Convention

7.2.1.2 *types* Element

Every programming language has its own definition of data type. The data type definitions used to describe the messages exchanged, so the client can interpret the data correctly. The *types* element provides a simple mapping from WSDL data type to programmatic data type.

```
<types>?
   <xsd:schema ... />*
</types>
```

Table 7.7 Types Element

WSDL data type	Java data type
xsd:base64Binary	byte[]
xsd:Boolean	boolean
xsd:byte	byte
xsd:dateTime	java.util.Calender
xsd:decimal	java.math:BigDecimal
xsd:double	double
xsd:float	float
xsd:hexBinary	byte[]
xsd:int	int
xsd:integer	java.math.BigInteger
xsd:long	long
xsd:Qname	javax.xml.namespace.QName
xsd:short	short
xsd:string	java.lang.String

Table 7.8 Standard data type mapping from WSDL to Java

For more complex data types, such as array of String, XML Schema definitions must be defined. In SOAP, data encoding is provided through the *encodingStyle* attribute in where developers can specify the rules for data types mapping. Using WSDL *types* attribute and XML Schema definitions are preferred if the development starts by describing the web service using the WSDL document.

7.2.1.3 *message* Element

Message element is an abstract definition of the transmitted data. A *message* element can have zero or many *part* elements, which specify the parameters or return values involved in a message similar to a function call.

```
<message name="nmtoken">*
  <part name="nmtoken" element="qname"? type="qname"? />*
</message>
```

Table 7.9 Message Element

7.2.1.4 *portType* Elemenent

A *portType* element defines a set of abstract operations that can be called and the corresponding message format for each operation. *portType* is equivalent to *interface* in object-oriented programming language. Each **operation** refers to an input message and output message.

Dependent on the sequence and order of input/output message, it defines the type or mode of operation. WSDL supports four basic operation types:

- **One-way**
  Exact one input message from client to the service
- **Request-response**
  Exact one input message followed by one output message. Optional *fault* element can be specified to indicate errors.
- **Solicit-response**
  The server sends a message and receives a response. The operation has one output message followed by one input message. Similar to *request-response operation*, an optional *fault* messages can be specified to encapsulate errors.
- **Notification**
  The operation has exactly one output message from the service

```
<portType name=”nmtoken”>*
  <operation name=”nmtoken”>*
    <input name=”nmtoken”? message=”qname” />?
  <operation>
</portType>
```

Table 7.10 One-way operation

```
<portType name=”nmtoken”>*
  <operation name=”nmtoken” parameterOrder=”nmtokens”>*
    <input name=”nmtoken”? message=”qname” />?
    <output name=”nmtoken”? message=”qname” />?
    <fault name=”nmtoken” message=”qname” />*
  <operation>
</portType>
```

Table 7.11 Request-response operation

```
<portType name=”nmtoken”>
  <operation name=”nmtoken” parameterOrder=”nmtokens”>*
    <output name=”nmtoken”? message=”qname” />
    <input name=”nmtoken”? message=”qname” />
    <fault name=”nmtoken” message=”qname” />*
  <operation>
</portType>
```

Table 7.12 Solicit-response operation

```
<portType name=”nmtoken”>
  <operation name=”nmtoken” parameterOrder=”nmtokens”>*
    <output name=”nmtoken”? message=”qname” />
  <operation>
</portType>
```

Table 7.13 Notification operation

Note that *portType* defines abstract notion, in the case of a request-response and solicit-response operations, a particular binding must be consulted to determine how the messages are actually sent (i.e. as a HTTP request/response or two HTTP requests).

7.2.1.5 *binding* Element

The *binding* element specifies a concrete protocol and data format for how the service is bound to the message protocol, particularly SOAP. There are two different binding styles, rpc and document. A SOAP binding have also encoded use or literal use [47].

```
<binding name=”nmtoken” type=”qname”>*
  <soap:binding style=”document|rpc” transport=”uri” />
  <operation name=”nmtoken”>*
    <soap:operation soapAction=”uri”? style=”rpc|document”?/>?
    <input name=”nmtoken”? > ?
      <soap:body parts=”nmtokens”? use=”literal|encoded”
encodingStyle=”uri-list”? namespace=”uri”? />
    </input>
    <output name=”nmtoken”? />?
    <fault name=”nmtoken” />*
  </operation>
</binding>
```

Table 7.14 Binding to SOAP

In addition, a common pattern *document/literal wrapped* is often used. Thus, we have totally five different binding styles (but the document/encoded style is meaningless and is never used). A common disadvantage with rpc binding style is that it cannot be easily parsed and validated (for example using a XML validator), because only part of the SOAP contents are defined in the XML schema. In document binding style, the content in SOAP is defined in a schema and therefore could be easily validated.

rpc/encoded:

Operation names in WSDL are directly mapped into SOAP and parameters include type encoding information.

WSDL:

```
<message name="myMethodRequest">
  <part name="x" type="xsd:int"/>
  <part name="y" type="xsd:float"/>
</message>
<message name="empty"/>
<portType name="PT">
    <operation name="myMethod">
        <input message="myMethodRequest"/>
        <output message="empty"/>
    </operation>
</portType>
```

SOAP:

```
<soap:envelope>
  <soap:body>
    <myMethod>
      <x xsi:type="xsd:int">5</x>
      <y xsi:type="xsd:float">5.0</y>
    </myMethod>
  </soap:body>
</soap:envelope>
```

Table 7.15 rpc/encoded SOAP binding

rpc/literal:

The WSDL is mapped similar to rpc/encoded except that the type encoding information is omitted.

SOAP:

```
<soap:envelope>
  <soap:body>
    <myMethod>
```

```
        <x>5</x>
        <y>5.0</y>
      </myMethod>
   </soap:body>
</soap:envelope>
```

Table 7.16 rpc/literal SOAP binding

document/encoded:
Nobody follows this style and is not WS-I compliant.

document/literal:
The operation names and type encoding information are omitted. The content SOAP can be validated because it is defined in a schema (emphasized in bold text in the table below).

WSDL:

```
<types>
 <schema>
  <element name="xElement"
type="xsd:int"/>
  <element name="yElement"
type="xsd:float"/>
  </schema>
</types>

<message name="myMethodRequest">
 <part name="x" element="xElement"/>
 <part name="y" element="yElement"/>
</message>
<message name="empty"/>

<portType name="PT">
 <operation name="myMethod">
   <input message="myMethodRequest"/>
   <output message="empty"/>
 </operation>
</portType>
```

SOAP:

```
<soap:envelope>
    <soap:body>
        <xElement>5</xElement>
        <yElement>5.0</yElement>
    </soap:body>
</soap:envelope>
```

Table 7.17 document/literal SOAP binding

document/literal wrapped:
In document/literal wrapped binding styles, the SOAP message looks similar as in rpc/literal. The difference is that the method name is mapped from WSDL wrapped element with a single input message's part. The message part is an element and has the same name as the operation.

WSDL:

```
<types>
 <schema>
  <element name="myMethod">
   <complexType>
    <sequence>
     <element name="x" type="xsd:int"/>
     <element name="y" type="xsd:float"/>
    </sequence>
   </complexType>
  </element>
  <element name="myMethodResponse">
   <complexType/>
```

SOAP:

```
<soap:envelope>
  <soap:body>
    <myMethod>
       <x>5</x>
       <y>5.0</y>
    </myMethod>
  </soap:body>
</soap:envelope>
```

```
  </element>
 </schema>
</types>
<message name="myMethodRequest">
 <part name="parameters"
element="myMethod"/>
</message>
<message name="empty">
 <part name="parameters"
element="myMethodResponse"/>
</message>

<portType name="PT">
 <operation name="myMethod">
  <input message="myMethodRequest"/>
  <output message="empty"/>
 </operation>
</portType>
```

Table 7.18 document/literal wrapped

7.2.1.6 *service* Elemenent

The *service* element defines a service as a collection of related ports or endpoints. The *port* element defines the endpoint by specifying an address for a binding.

```
<service name=”nmtoken”> *
  <port name=”nmtoken” binding=”qname”> *
    <!-- extensiblility element -->
  </port>
</service>
```

Table 7.19 Service Element

7.3 UDDI

Universal Description, Discovery and Integration (UDDI) [48] is a protocol that implements the directory of services for Web services. It enables service provider to publish and register web services to a central registry. Service consumers can search/discover available services.

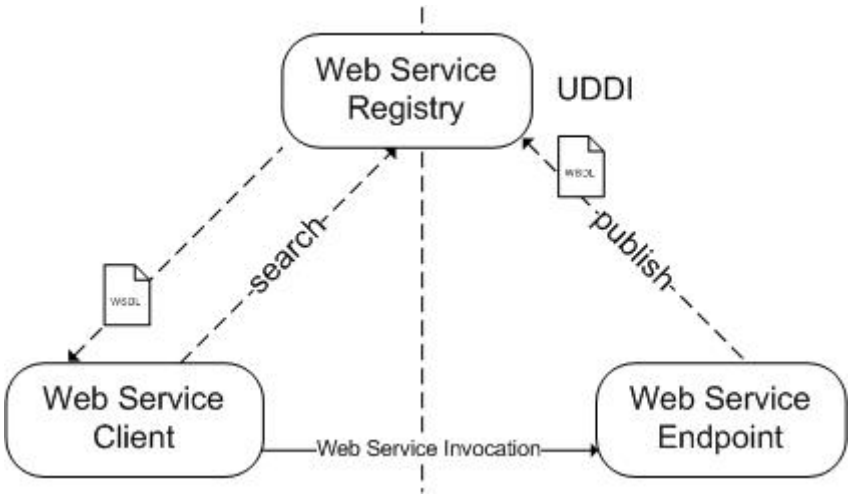

Figure 7.6 UDDI

Mobile Home Access does not publish the available web services in a registry, but the WSDL document describing the service is provided (by manually download from the Internet). UDDI is from now on not referred.

PART II:

DESIGN

Chapter 8 Requirements

In every software development project, the identification of the system requirements is an important process. This includes high-level view of the specified system, i.e. identification of users of the system, requirements of the system and its users, and the challenges associate with these requirements.

From the use case view with UML use case diagrams, we deduce the functional requirements. We may also identify non-functional requirements. When requirements are identified, we can move to the next milestone. This chapter describes the first step of the development process to identify the problem domain by using use case diagrams to specify system's requirements.

8.1 Scenarios

The best way to understand the system is to figure out how the system works and how it applies to the real world/application without concerning about the implementation and technical issues. The scenarios below are examples of the usage of the system. In fact, the usages of the final product could be beyond our predictions. In the following scenarios, we assume that the home network is configured correctly and the Mobile Home Access is installed both on home network (i.e. server) and on remote clients.

Remote access (from PC)
At work, Ola realizes that documents and files he worked on last night on the desktop computer at home were not copied to his laptop. He connects to the home network via MHA and downloads the files needed. He also mounts his user directory as a local hard disk so he can work directly without copying back at the end of the day.

Remote access (from mobile phone)
On the way home from work, Ola wants to listen to a song that is not in his mp3 player. Because it has limited storage, he just can simply copy all his music from the desktop computer to his mp3 player. He also has another problem because the mp3 player does not have any kind of network connection except Bluetooth. Ola uses a workaround method using his mobile phone to connect to MHA and then transfer it to the mp3 player via Bluetooth.

File sharing between users
When he got home, he gets a phone call from Kari. She asks Ola to send her the pictures that they took during their vacation in Paris. She cannot access it because they are in Ola's private directory. He moves the pictures to the public share folder, which every family member has access.

Remote storage
Later that night, Ola went to his friend's birthday party. Using his camera on the mobile phone, he took some pictures. Unfortunately, the mobile phone went out of space. But he knows he can connect to his home network and transfer all the pictures from the mobile phone to his personal folder. Thus, the system functions as a remote storage of personal files and documents.

8.2 Identification of actors

We start the use case view by identifying the actors of the system. From figure 1.1 we can see that the system consists of users, terminals and network components/devices. Dependent on the level of abstraction, the actors could be any of these components or other artifacts.

8.2.1 Terminals

Users communicate/interact with the system and access the functionalities through the (graphical) user interface provided by the application on the terminals. The terminal then sends the requests to the system and receives responses. The ideal case is when the functionalities of a system are independent of the type of terminals. Unfortunately, limitations on the terminals prevent the system from providing full functionality to all terminals and thus users.

Even though clients can be grouped in many different categories (such as personal computers, laptops, and mobile devices), in this thesis the devices or terminals are grouped into two main groups:

- *Rich clients*
- *Restricted clients*

8.2.1.1 Rich Clients

Rich clients are clients with high processing power and resources (such as hard disk space, memory, and screen resolution). For such clients, the system can provide full functionality.

Note:
Even though clients may have all resources available, we must also take into account the restrictions in the operating system. The operating system may have access control enabled for certain services that prevent users to perform special operation.

Examples of such clients are:

- Personal computer/PC
- Laptop

8.2.1.2 Restricted Clients

Restricted clients have low processing power and resources. The clients may have limited screen size, memory and also navigation buttons. Computers with restricted resources and capacity for users fall also into this category.

- PDA
- Cell phones
- Smart phones
- Thin clients

Physical Restrictions
Typical restrictions on mobile devices are the dimension of the screen, navigation facility, and the keypad.

Technical Restrictions
These restrictions include the size of storage (memory, internal storage), processing power.

Users' Restrictions
On multi-user terminals, such as a computer, a user has restricted access and limited rights to computer's resources. On thin clients[7], those you commonly find on the library or school, users may not be able to install new hardware and/or software applications.

8.2.2 Types of Users

Similar to various terminals, there are also different types of users. Users can be categorized into:

- **Users** (Regular users)
  Users with limited access and functionalities
- **Administrators** (Power users)
  Users with extended access and functionalities to be able to alter configuration and settings of the target home network.

8.3 Use Cases

Use case modelling is an excellent method to capture the functional requirements of the system. The use case diagrams are easily understood and it is not only used by programmers but also by the project leaders to address the system functionalities in an simple way.

8.3.1 Initiation of service

Before a user can access the service on the system, he/she must have a successfully established connection with the server application. During this phase, the user must authenticate his/her identification by logging in with username and password.

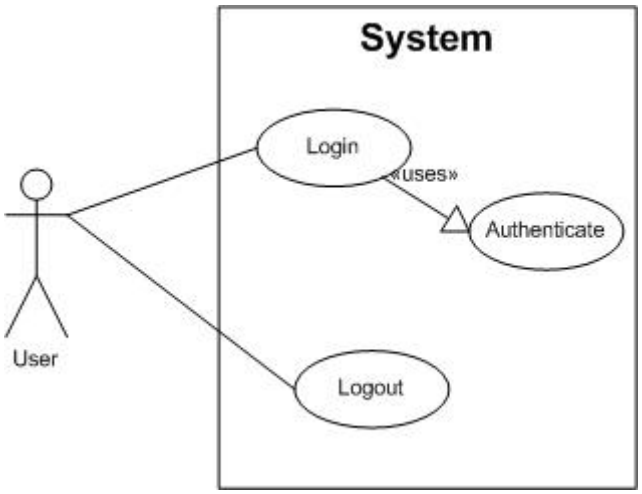

Figure 8.1 Login/logout

When users step away from the terminal, it is preferred to logout to end the current session. A user may risk other users to be able to reestablish the active connection/session and access private files and documents if the session is not destroyed. The management of the session should prevent this by setting the session expire time lower or to regenerate the session id.

[7] A thin client is a computer in client-server architecture networks which has little or no application logic. The processing activities is primarily dependent on the central server.

Security:
The login process should be performed in a secure connection to prevent the username and password from being intercepted. Otherwise, a strong password with combination of lower-case and upper-case letters, and numbers is preferred. Also avoid sending passwords in clear text.

8.3.2 Administration and Configuration

On most software applications there are administration interfaces which can be accessed by administrators and users with authorized rights. Regular users normally have the ability to customize their private settings and/or application's behaviors.

In our system, the administrator can change all the system settings and user's preferences. The administration tasks are grouped into following use cases (as depicted in the use case diagram in figure 8.2):

- **Users administration**
  Examples of task: enable/disable a user account, setting other users to administration role
- **System settings and configuration**
  Examples of task: start/stop services, install/uninstall services
- **Usage statistics**
  There are several cases an administrator may consult the system's logs (such as access and error logs) and usage statistics of the system, for example when the system misbehaved or is not working properly.
- **Administrate resources**
  Administrators can set permission settings for all *shares* and resources such as printer, etc. Note that, an administrator must also have administrator account/role on every computer on the networks as on Windows Workgroups do not have a centralized user database.

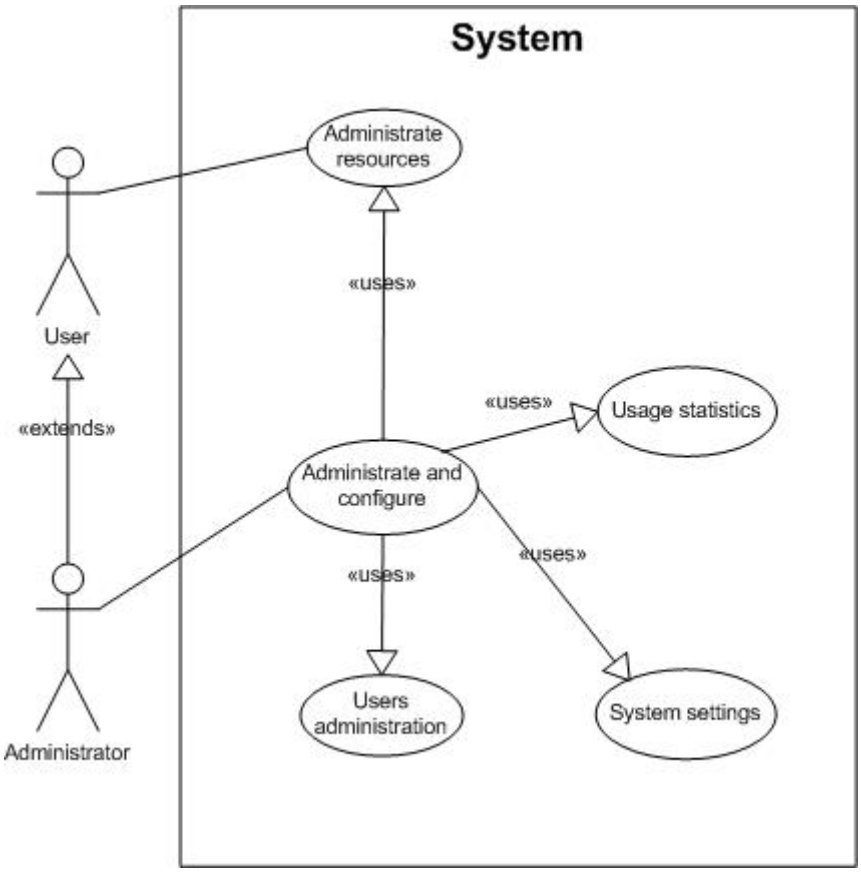

Figure 8.2 Control and management use cases

8.3.3 Services

Accessing files and documents at home networks consists of following functionalities:

- **Listing/Browsing**
  As in Network Neighborhood/My Network Places, users should be able to view computers/servers on a workgroup as well as shares available for each server.
- **View resource properties**
  Similar to Windows Explorer, when user right-clicked on a file/directory icon and choose *properties* on the popup menu, a detail of the file/directory will appear. This functionality is also available for other devices such as printer, scanner, and etc.
- **Mount/unmount**
  Users should also be able to *map network drive* to a share, and thus working on it in the same way as other local resources on the computer.
- **Download/upload**
  For users with slow Internet connections, it could be wise to download a file or directory to the local hard drive instead of working on directly. When users finished working on it, they can upload it to the server.
- **Modify**
  Users can also perform file system operations such as renaming, deleting, and create new files or directories. The system should also enable many users to access and modify the same resource as the same time by using resource lock and unlocking features, in addition to commit any changes or synchronize and update to the current state.

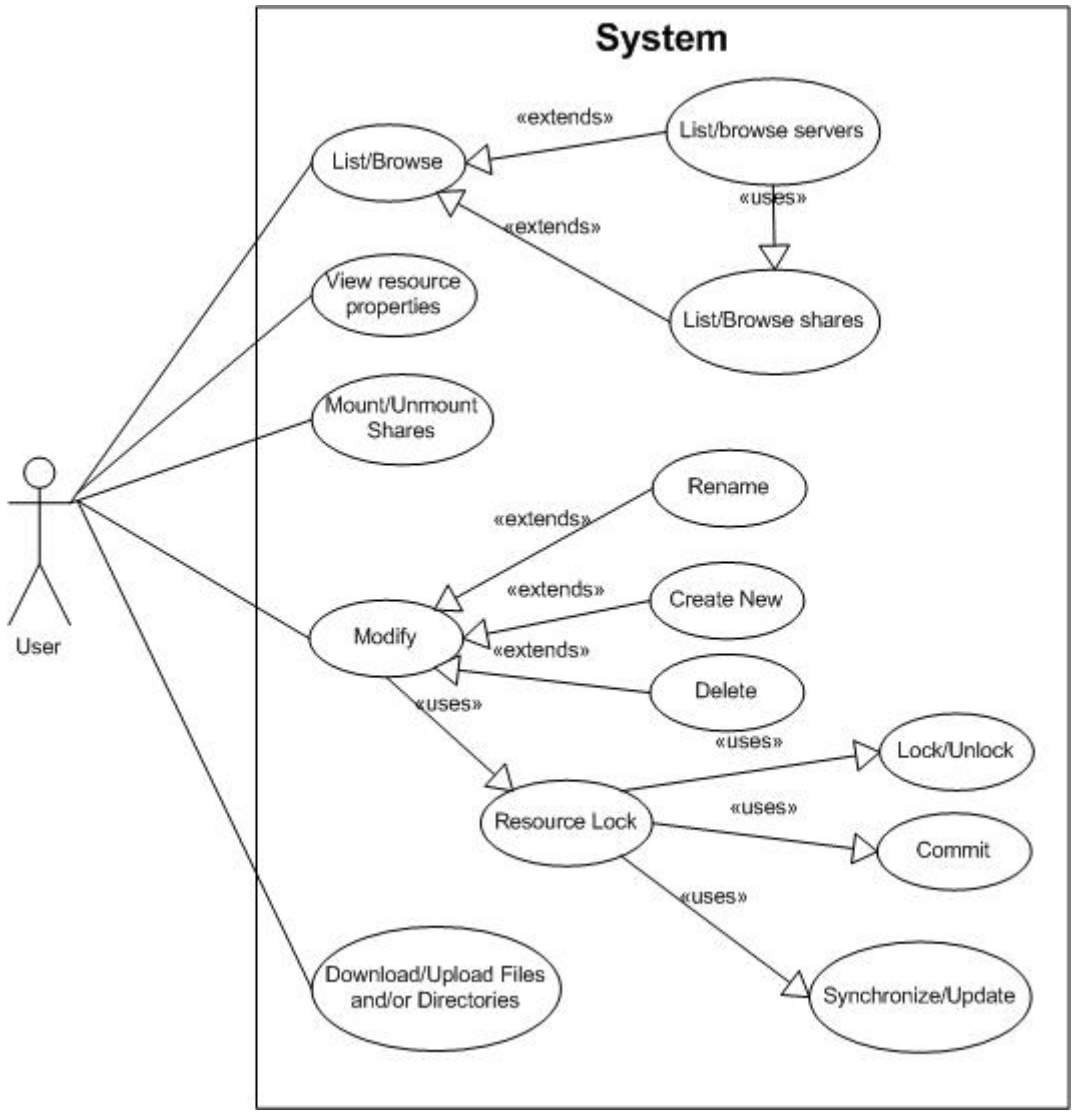

Figure 8.3 File system use cases

Devices such as printers that are connected to the home network can also be shared. In the future, other devices can also be connected and shared on the network using technology such as Universal Plug-and-Play (UPnP)[8].

[8] UPnP is a technology that simplifies the implementation of networks and connect easily to the devices[49].

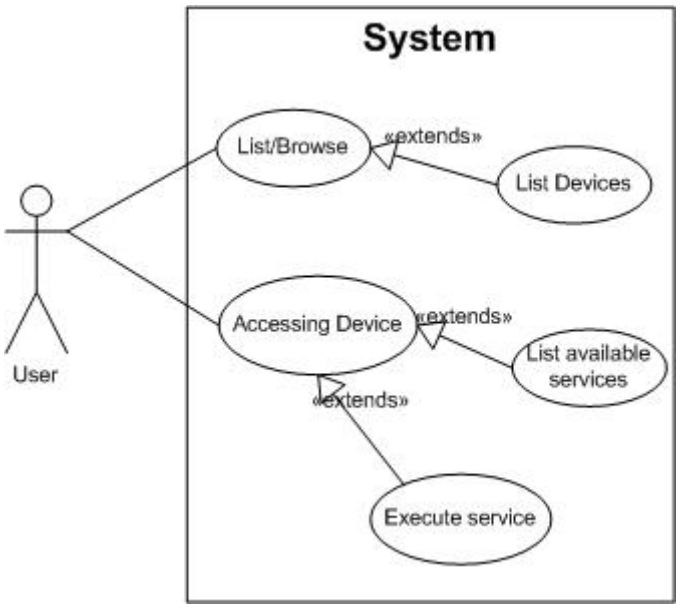

Figure 8.4 Accessing devices

For network devices, the system will list them together with the list of servers on the network. When devices are physically connected and shared from a specific computer (server), the device will come up as the computer's resource (for example a printer).

Accessing a device should be as simple as possible, because for each device there are a different set of services it offers. Thus, the use cases consist of listing the services available and then execute them. There are of course restrictions and requirements to execute a service, and therefore we may not be able to use a service even though it is available.

8.4 Functional Requirements

From the previous use case diagrams, the system's functionalities are described as how end-users will expect the system to function. We will in this chapter and the next chapters, describe the requirements of the system in more technical form, e.g. functional and non-functional requirements.

Definition of functional requirements:
Functional requirements are what a system should be able to do, and the functions it should perform.

FR1: Access Control/Authentication
Authentication is one of the most common functionalities in modern IT-systems. It is important that the system authenticates users, and functions as a gatekeeper to prevent access by unwanted users. The most commonly used user's identification is using a username and a password. The services of the system are only available after user authentication.

FR2: Authorization
In a multi-user's environment, the system must differentiate between users, i.e. a user can have extensible rights to perform administrative operations. Access control prevents other (home-) users from performing unauthorized operations (such as changing system settings). This is also very important for critical services provided by for example alarm system, consumer electronics and other network devices such as router. Without access control, all authenticated users may execute these services and therefore may cause unpredictable results.

FR3: Users and Terminals Transparency
In Windows workgroup, there is no central repository of user information (passwords, profiles, etc). Each computer has its own user database which keeps the information about their local users. The system must provide a mechanism to register and map remote user to the local user account.

Terminals on the home network normally have private addresses (i.e. IP address) which are not visible outside the internal network. Therefore, the system must support address mapping and routing.

FR4: Administration and Configuration
The system should provide an administrative interface that enables the administrator to change its settings and configuration. Regular users can also modify settings for their private shares and settings for application behavior.

FR5: Listing/Browsing Services and Resources
Users will be provided a list of available resources so they can navigate/browse (for example in directory to show the contents of the current directory). The list should also, if possible, show the resource type (such as a file system resource or device).

FR6: Download and Uploading
Users should be able to download single or multiple files at once, and may also download a whole directory to the mobile host. Resuming a transfer can be supported in case of connection problems.

FR7: Mounting and Un-mounting Shares
Mapping a share as a local hard drive or local resource should be supported for rich clients, i.e. computers. Using this method, users work on the resources similar to the way they are at home.

FR8: Files and Directories Manipulation
Operations on files such as editing, deleting, creation of new files and directories, and other common file operations in the operating system should also be supported as well. On restricted terminals (e.g. mobile phones), some of these operations cannot be done because of their limited resources. For example, editing a Word-document in a mobile phone requires that the device's operating system must have built-in support for such operations and file formats.

8.5 Non-functional Requirements

Definition of non-functional requirements:
Non-functional requirement is a requirement that defines a system's properties and constraints.

The non-functional requirements should be specified early in the development process, because postponing them to end phase may result the whole system to be rewritten to meet specific quality requirements.

NF1: Easy to Setup, Use and Configure

One of the most important human factors, as far as software application is concern, is the ease to setup, use and configure the application to suit individual's needs. The system should provide wizards and user-friendly graphical user interfaces (GUI) wherever the system requires user intervention. The application should display informative messages and avoid using incomprehensible text/number that causes users' frustration instead of being informative.

NF2: Performance
Performance has become more and more important by end-users as the computer's processing power increases. Performance measurements are normally speed (how fast an operation is executed), throughput (how much data it can process), and response time (the time between a request starts till the response is received).

It is nearly impossible to meet all performance requirements without conflicting/sacrificing other. In networked applications, i.e. client-server architecture, the response time is of great importance.

NF3: Security
When data pass through the network, they are exposed by many security threats. The data may be eavesdropped and replayed by other users. The system should prevent common attacks by using cryptographic functions such as encryption/decryption for confidentiality and secure hash functions for data integrity.

NF4: Availability
The system is useless if it keeps breaking-down. Error detection and error handling should be available to ensure the system from breaking-down. In addition, recovering from system crash is highly recommended.

NF5: Extensibility
New functionality and services should be easily integrated into the existing system. Thus, the system architecture must be modular and extensible to ease the deployment process.

NF6: Platform- and Network Infrastructure Independent
The system should be independent of platform and network infrastructure. Designing a system that depends on implementation detail of the underlying architecture is hard to maintain and should be avoided if possible.

NF7: Documentation
The system should provide documentation and user's guide for regular users, as well as for system and service developers.

Chapter 9 System Architecture

One of the common decision at the early stage of development is that if the system could use existing architecture and framework (to save time and efforts), or must it be created from scratch. Normally it is not necessary to start from scratch as there are many open source or free technologies that can be used as the system's backbone and basis building block. In this chapter, we describe the main problems of existing solutions in greater details and the architecture Mobile Home Access is based on. It is argumented why the selected architecture is suitable for the specified system. Then we will take a look into the MHA architecture and its design.

9.1 Networked File System Restrictions

The networked file system, CIFS, is a widely used protocol that was designed to enable computers to share files and other resources with computers on a local area network. But to enable it to work in wide area networks (WAN), users must create separate servers (WINS/Domain Server). Most companies and large organizations deploy it on top of a VPN solution, but for regular users the solution is cumbersome and difficult to accomplish in terms of technical insight.

The system is required to be network infrastructure independent and without changing network architecture. We will describe why this requirement is hard to achieve because of firewall and NAT.

9.1.1 Problems with Firewall and NAT

The firewall typically blocks services that run on ports below 1024. This implies that CIFS services which run on ports 137, 138, 139 and 445 are likely to be blocked. We may open these ports, but this leaves security holes in our system. In addition, CIFS protocol may send unencrypted authentication information and data which can be intercepted by others listening on the traffic.

Assume that security is not the problem and these ports are opened, how can we route requests to the right destination since each computer in the home network is running the same services at the same ports. Using NAT port forwarding, only one service at a computer can map to a specific port. We may specify different ports to every services (for example port 1137 maps to computer A on port 137 and 2137 maps to computer B port 137). This will be too messy and technically impossible to accomplished as it is not possible to specify different ports in CIFS clients.

9.2 The Architecture of MHA

The architeture of Mobile Home Access is based on Web Services, an implementation of Service-Oriented Architecture (SOA).

Web Services can bind to many widely used transport protocols, but the most common is by using HTTP transport binding. To allow connection to the web server, the port for the service (normally, port number 80 or 8080) must be opened in the network firewall. With port forwarding, web services requests can be forwarded to the Mobile Home Access server, which runs as a web application in a web server with Web Service support. This solves the

problem of firewall by binding to a single HTTP port (typically port number 80 or 8080), instead of opening all ports associate with CIFS services.

The Mobile Home Access web service then routes these requests internally to the correct destination or process these requests if the services is on the local computer. This solved the traversal of NAT-enabled router and the routing issues. Also, to ensure that the CIFS service will run properly, check that the ports (137, 138, 139, 445) are opened for incoming connection in personal firewalls.

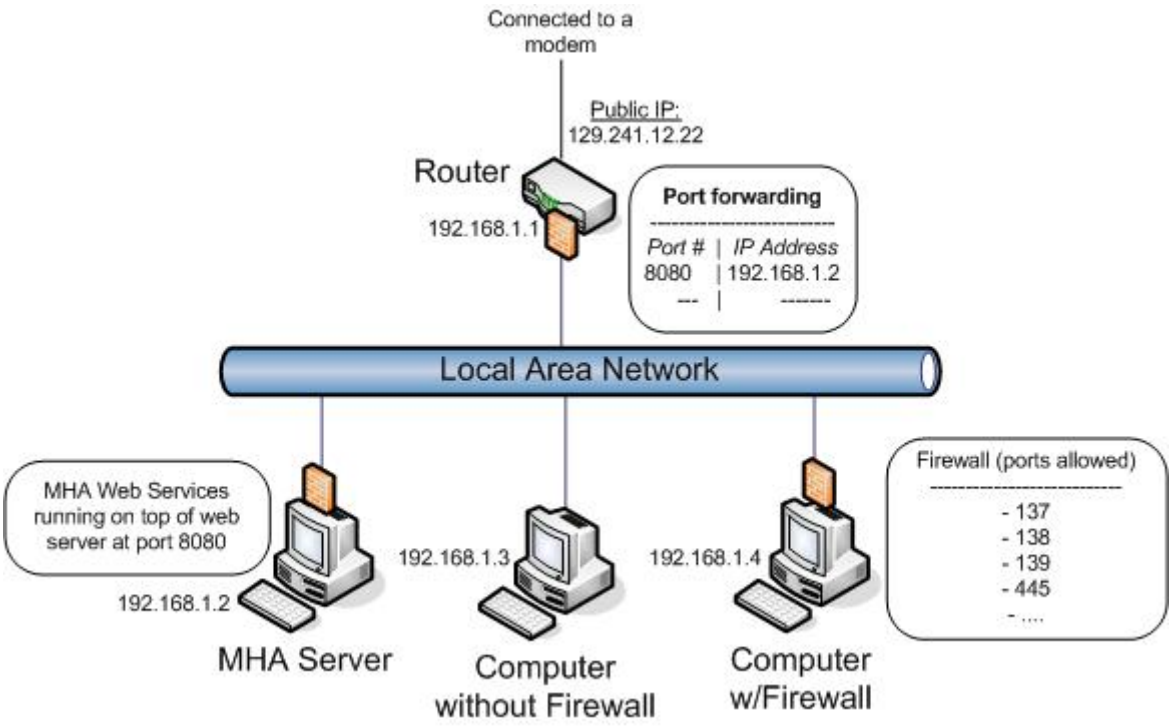

Figure 9.1 Example of Home Network configuration

Above is an example of a home network with private IP address range from 192.168.1.1 to 192.168.1.254[9] and a public IP address 129.241.12.22. These IP addresses are fictive and may differ for the actual home network. The implemenation of web services for Mobile Home Access is running on top of a web server which is located in computer with an IP address, 192.168.1.2. In the router, an entry to the port forwarding table to route incoming requests/packets to ports 8080 to be forwarded to MHA Server. For all (personal) firewalls in home networks, ports for CIFS services is opened as illustrated for computer 192.168.1.4.

9.2.1 Modes of Operation

Various types of terminals will be connected to Mobile Home Access, thus "*one size doesn't fit all"*. As described earlier, terminals have restrictions.

For restricted clients, the system's functionalites is mapped to SOAP messages. The server at home network will construct and instruct an CIFS client to handle these requests. This reduces the packet size to SOAP messages instead of CIFS packets. This mode is called *Reduced-Mapping Mode* [50].

[9] 192.168.1.0 and 192.168.1.255 are reserved for other purposes and could not be used as IP address for a network node

For rich clients, which have full system functionalities, there are two different solution. We can use the same approach as in Reduced-Mapping and map all CIFS commands to SOAP messages. But this is not preferrable, because of the number of different CIFS command we must support. The system will have difficulty to maintain the consistency when the protocol changes, for example new command is added to or remove from CIFS. A better approach is to encapsulate CIFS packets in an SOAP message in a similar way as tunneling, and hence the name *Tunneling Mode*.

9.2.1.1 Reduced-Mapping Mode

The Reduced-Mapping Mode will not support following service:

- Directly mounting and un-mounting share
- Directly access and use devices

The modified use case diagram for Reduced-Mapping Mode is as follows:

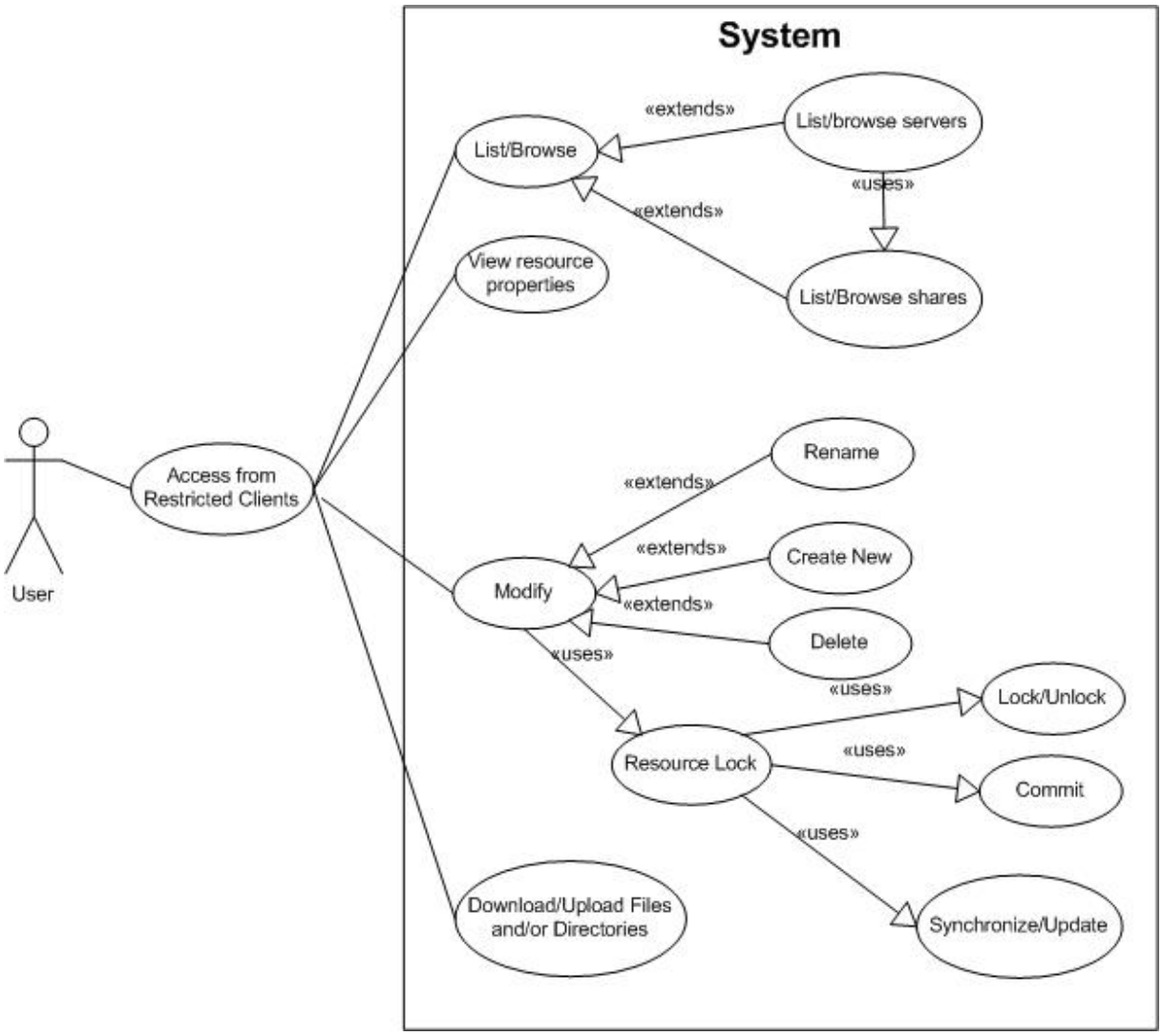

Figure 9.2 Reduce Mapping Mode use cases

Some file operations such as open cannot be executed in all devices; this is because the file type is not supported by the device operating system. For some devices (such as PDA), a third-party library can be installed and used to open such file. Since the device can access the

file system directly, editing a file requires the clients to download the file before it can be edited.

9.2.1.2 Tunneling Mode

In Tunneling Mode, CIFS packets are encapsulated in SOAP messages. The services in figure 8.3 and other services (except control and management) that CIFS supports will be available by tunneling. All CIFS requests to the home network are captured and exposed by sending to the Mobile Home Access client, which will encapsulate it into SOAP messages and send to the server.

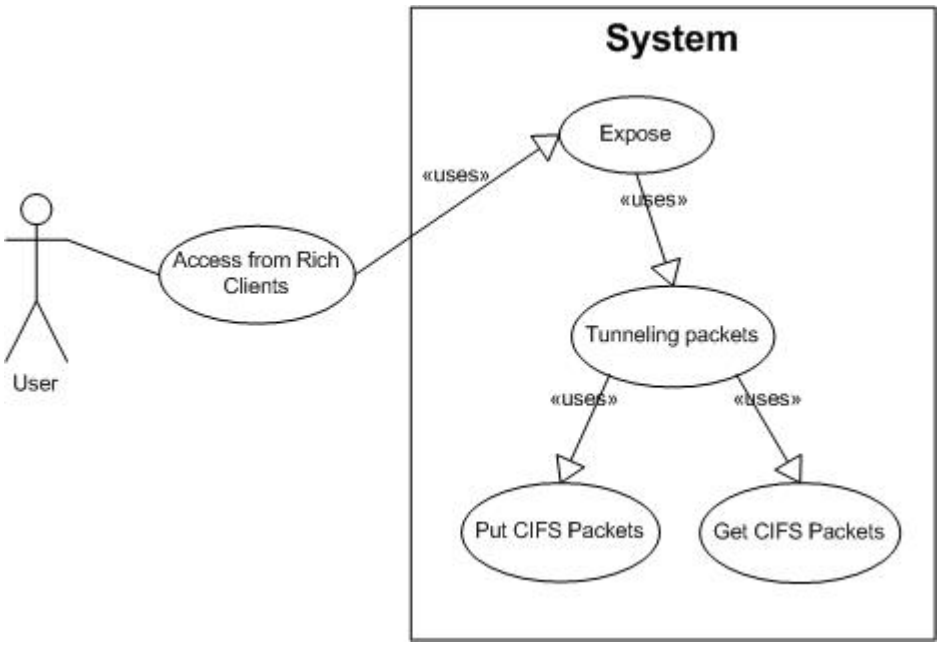

Figure 9.3 Tunneling Mode use cases

Expose
In this use case, all CIFS requests are intercepted.

Tunneling packets
Intercepted CIFS packets is then encapsulated into a SOAP message. SOAP messages received is then decapsulated and sent to the client.

Put
Sending the request (SOAP message) to the server.

Get
Receive the responses (SOAP message) from the server.

Chapter 10 System Design & Analysis

This chapter's main goal is to translate the use cases in the previous chapter into a system design. We have studied the use cases and defined the requirements of the system, in addition to the modes of Mobile Home Access. In this chapter, we'll design the system operating in Tunneling Mode.

10.1 System Interfaces and Boundaries

To identify the interactions between components of the system, we address and separate the boundaries of the system. From the figure 1.1, we can identify the boundaries of the system. Two significant boundaries are identified; the home network and the remote network (where remote client is connected to). Because the system is required to be independent of network architecture, Internet and ISPs are not considered.

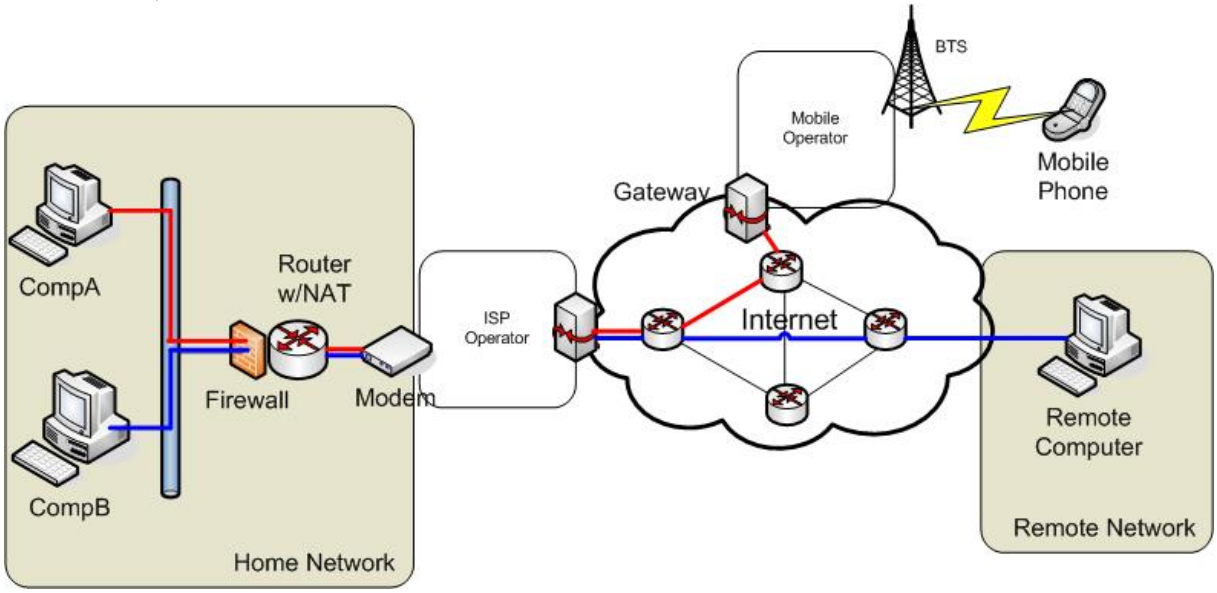

Figure 10.1 System Boundaries

Home networks and clients at the remote network are connected via the web service interface. Local resources are then accessed via the local interface. A resource is either on the computer that runs the Mobile Home Access server or on other computers in the home network.

On remote computers, a CIFS client accesses resources at home network by first sending packets via the virtual network interface to Mobile Home Access client, which will forward them to the home network via the web service interface.

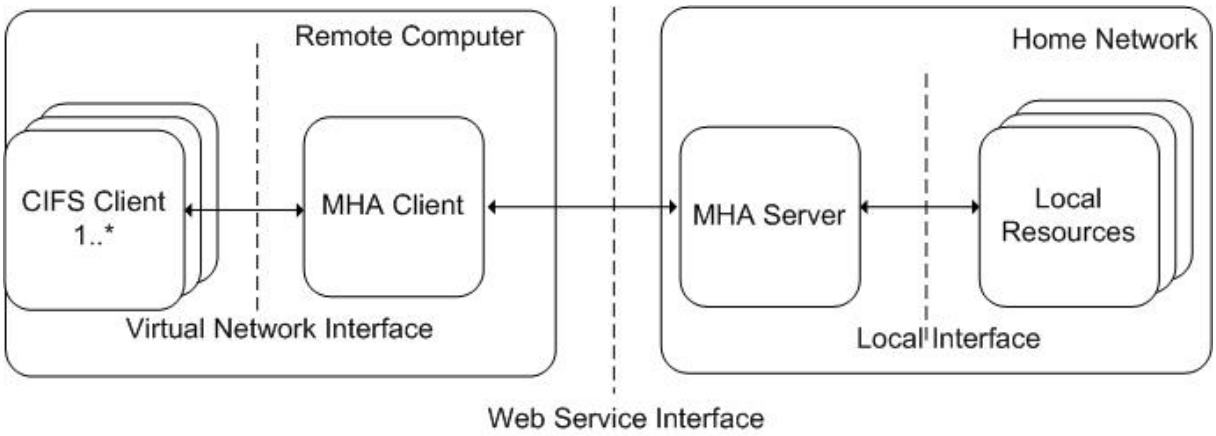

Figure 10.2 Interfaces

10.2 Sequence Diagrams

The use cases for Tunneling Mode are then translated into sequence diagrams (one of UML interaction diagrams), which model the logic flow of an operation.

10.2.1 Login and Logout

This sequence diagram (figure 10.3 below) is for the login and logout process. When the client application has started up, a login screen will be displayed and asks the user to type in the username and password, and choose operation mode (i.e. Reduced-Mapping Mode or Tunneling Mode) as well. The information will be send to the server in a SOAP message. The server responds with a Boolean value (true or false), which indicate the successful status of login process and session identification in case of successful login. Upon failure, the login screen will reappear so users can re-enter the correct username and password.

For logout process, the server cleans up resources in use by the session and then destroys the session.

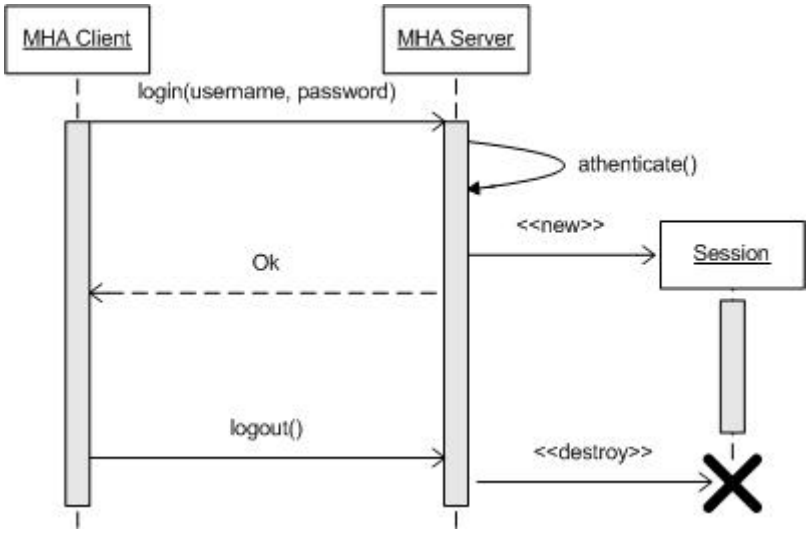

Figure 10.3 Login and Logout

10.2.2 CIFS Services

CIFS services are exposed via *put* and *get* use cases. All CIFS packets from the client side is forwarded to Mobile Home Access client. The MHA client functions as a proxy, which

encapsulates the packets into SOAP messages and sends them to the server via web service interface.

When the MHA server receives a SOAP message from a client, it decapsulates the SOAP message. The MHA server creates a new CIFS client which connects and sends the packets to the target CIFS server. When a response is received, the packets are sent back to the MHA clients in the similar way.

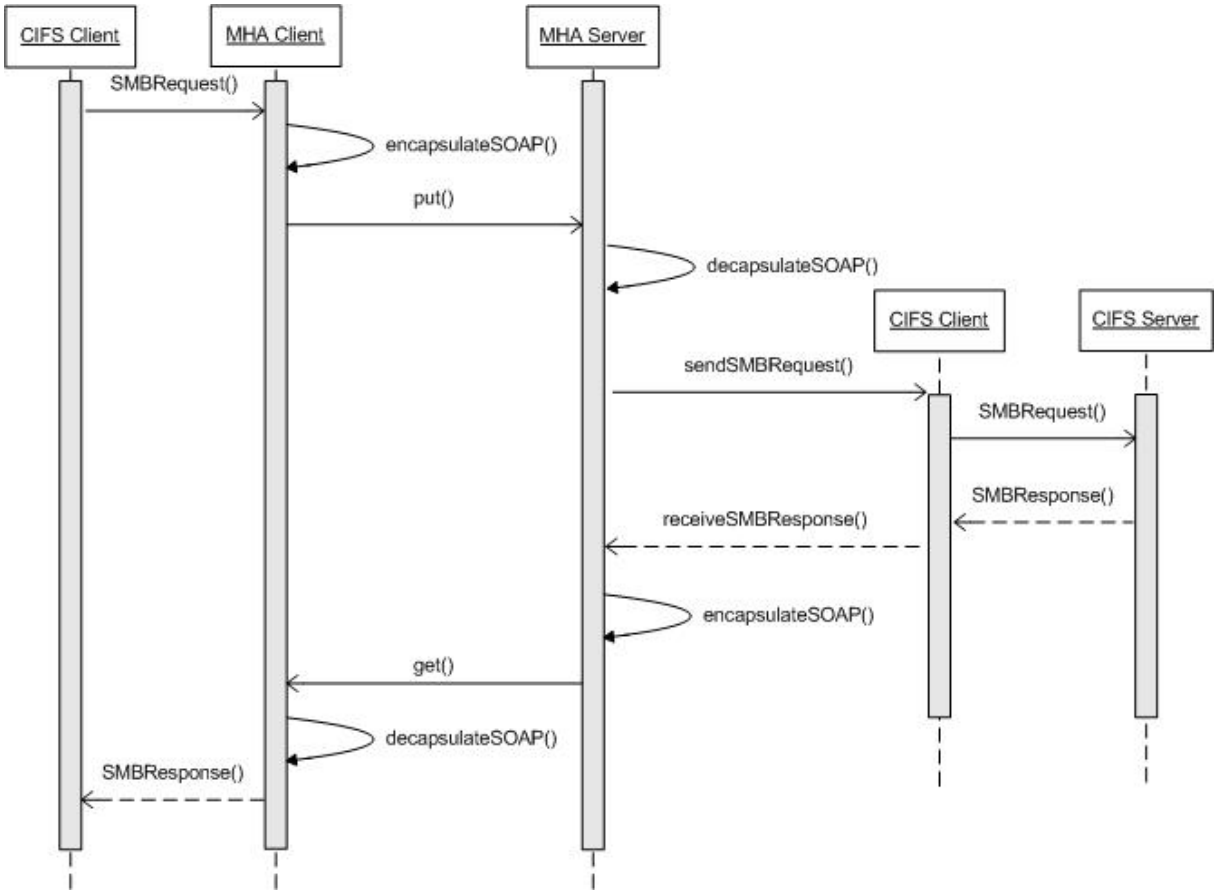

Figure 10.4 CIFS Services

10.2.3 Administration

Upon request from user, the MHA client will show settings and configurations via a graphical user interface. The available settings and configurations may differ based on the role of the user. The changes will be registered and sent to MHA server if these settings are global, i.e. permanent change of application or the behaviour of the system. The server then responds with a boolean vaule to the client, which then notifies the user if the changes have been made or not. For local settings, such as application appearance, the MHA client simply notifies to the user when the settings are saved in the local configuration file. The sequence diagram for general administration tasks is shown below in figure 10.5.

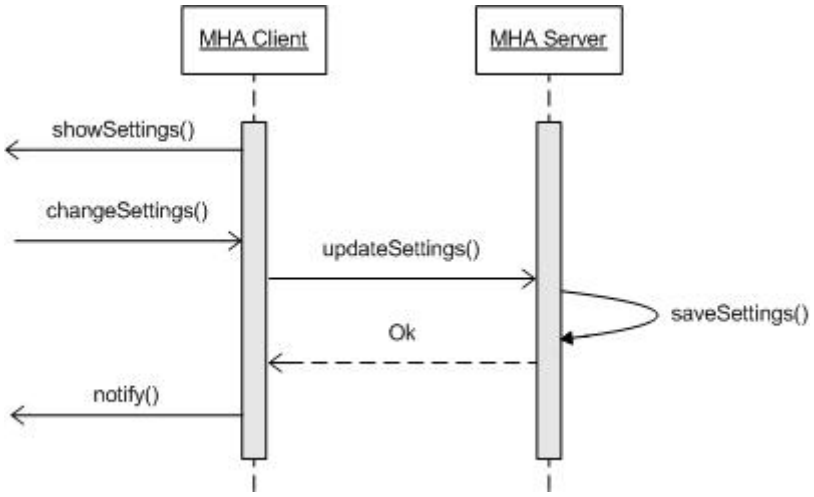

Figure 10.5 Administration

10.3 Class Diagrams

Class diagrams model class structure (classes with their attributes and operations) of the system. In class diagrams below, many associations among classes and interfaces has been left out on purpose to provide a better view of the system.

10.3.1 Client-Side

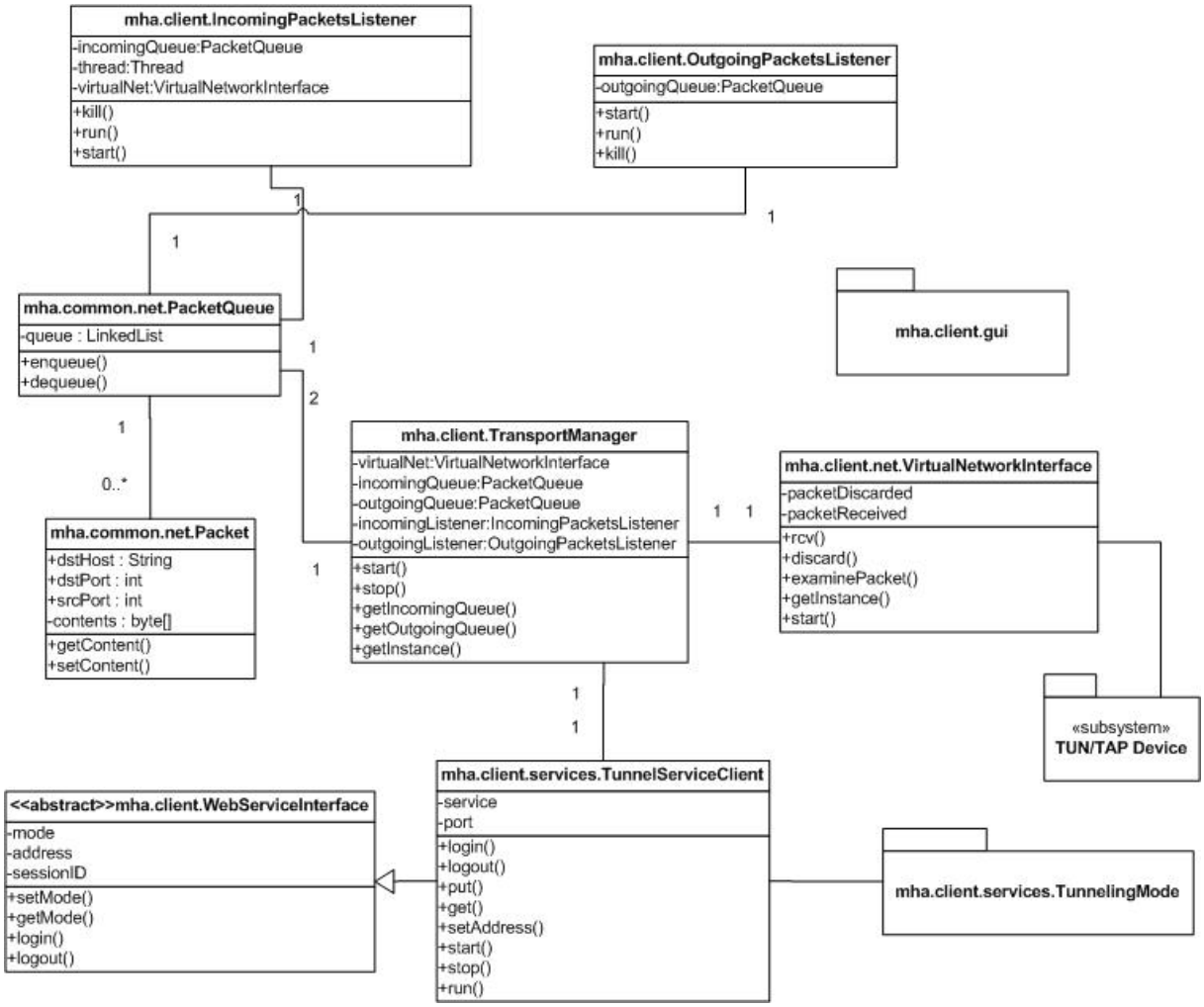

Figure 10.6 Client-side Class Diagram

Class/Interface	Description
Packet	This class represents a CIFS packet and contains attributes about where the packet is destined (dstHost, dstPort) as well as to which port (srcPort) on the remote computer where the responses will be sent.
PacketQueue	This class implements a FIFO queue for CIFS packet (instance of Packet class). Two common methods for a queue are adding a packet to the queue (enqueue) and remove from the queue (dequeue)
IncomingQueueListener	This class implements the *Runnable* interface, and listens to incoming PacketQueue and processes packets as they arrive, i.e. sends to CIFS clients via VirtualNetworkInterface
OutgoingQueueListener	This class is similar to IncomingQueueListener, except that it listens to outgoing PacketQueue and sends packet to Mobile Home Access server via TunnelingServiceClient
TransportManager	This class is a controller class which manages the queues and their listener
VirtualNetworkInterface	This class interacts with the TUN/TAP device using JNI to send and receive CIFS packets to and from CIFS clients on the remote computer
WebServiceInterface	This is an abstract class that contains common functions/methods in Mobile Home Access web services, such as login and logout.
TunnelServiceClient	This class creates endpoint to communicate with the web service and implements functions to invoke web service's operations.
mha.client.services.TunnelingMode	This package is generated from the WSDL document and contains classes and interfaces which provides client's stub to call the web service.
mha.client.gui	This package contains GUI components, such as frames and panels which are visible for users.
TUN/TAP Device	This is the native (i.e. in C programming language) implementation of JNI functions in VirtualNetworkInterface class which communicates with the TUN/TAP device.

Table 10.1 Description of client-side classes and interfaces

10.3.2 Server-Side

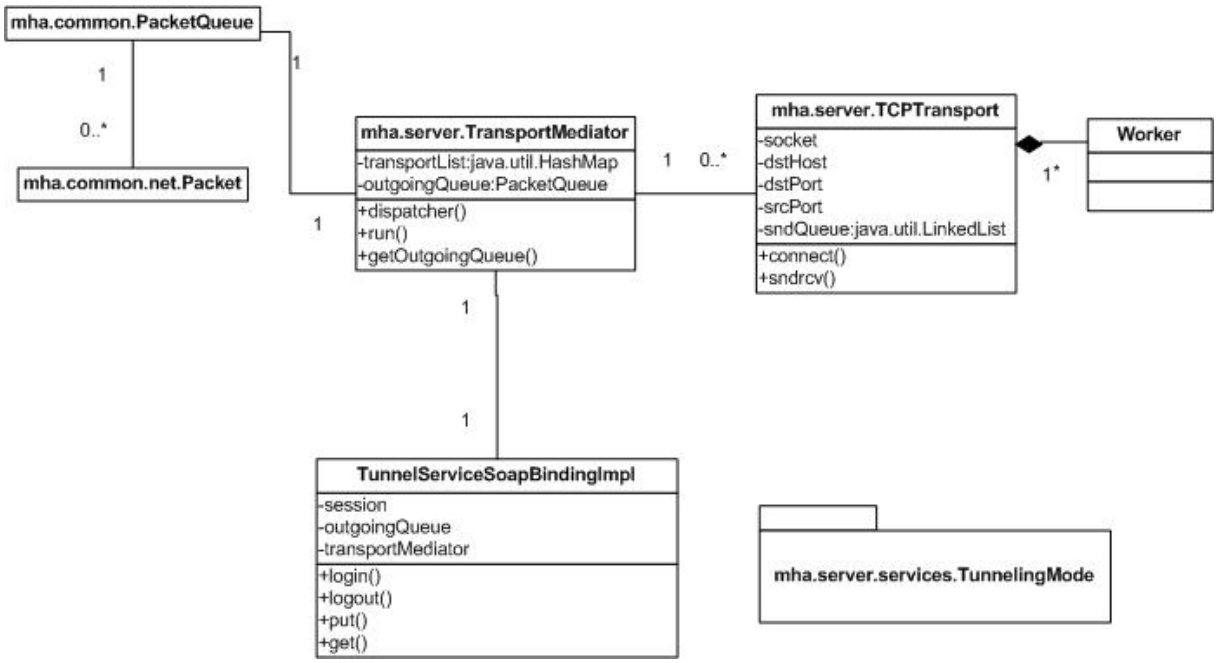

Figure 10.7 Server-side Class Diagram

Class/Interface	Description
TCPTransport	This class sends and receives CIFS packets. via internal *Worker* class, to and from CIFS server on the home network.
Worker	This is a private class in *TCPTransport* class and implements the *Runnable* interface to create a thread for handling the actual communication to CIFS server.
TransportMediator	This class dispatches packets to TCPTransport instance
TunnelingServiceSoapBindingImpl	This class is part of the *mha.server.services.TunnelingMode* package and implements the Tunneling Service
mha.server.services.TunnelingMode	This package contains Java files generated from WSDL. The implementation template is generated in *TunnelingServiceSoapBindingImpl*.

Table 10.2 Description of server-side classes and interfaces

10.4 Deployment Diagrams

A deployment diagram shows the static view of the run-time configuration of processing nodes and components that run on them. The diagram includes hardware for the system, the software installed on those hardware and/or middleware used to connect components, hardware and/or machines together.

On remote computers, a CIFS client (either provided by the operating system itself or other applications) accesses CIFS servers on the home network via the virtual network interface and Mobile Home Access client. The request from the MHA client is processed by the Axis SOAP engine which then invokes the implementation of the Mobile Home Access web service. The web service then sends requests from remote clients to either the CIFS server on the same machine that runs Mobile Home Access web services (denoted as Server Computer in the figure below) or to other CIFS servers on the local area network.

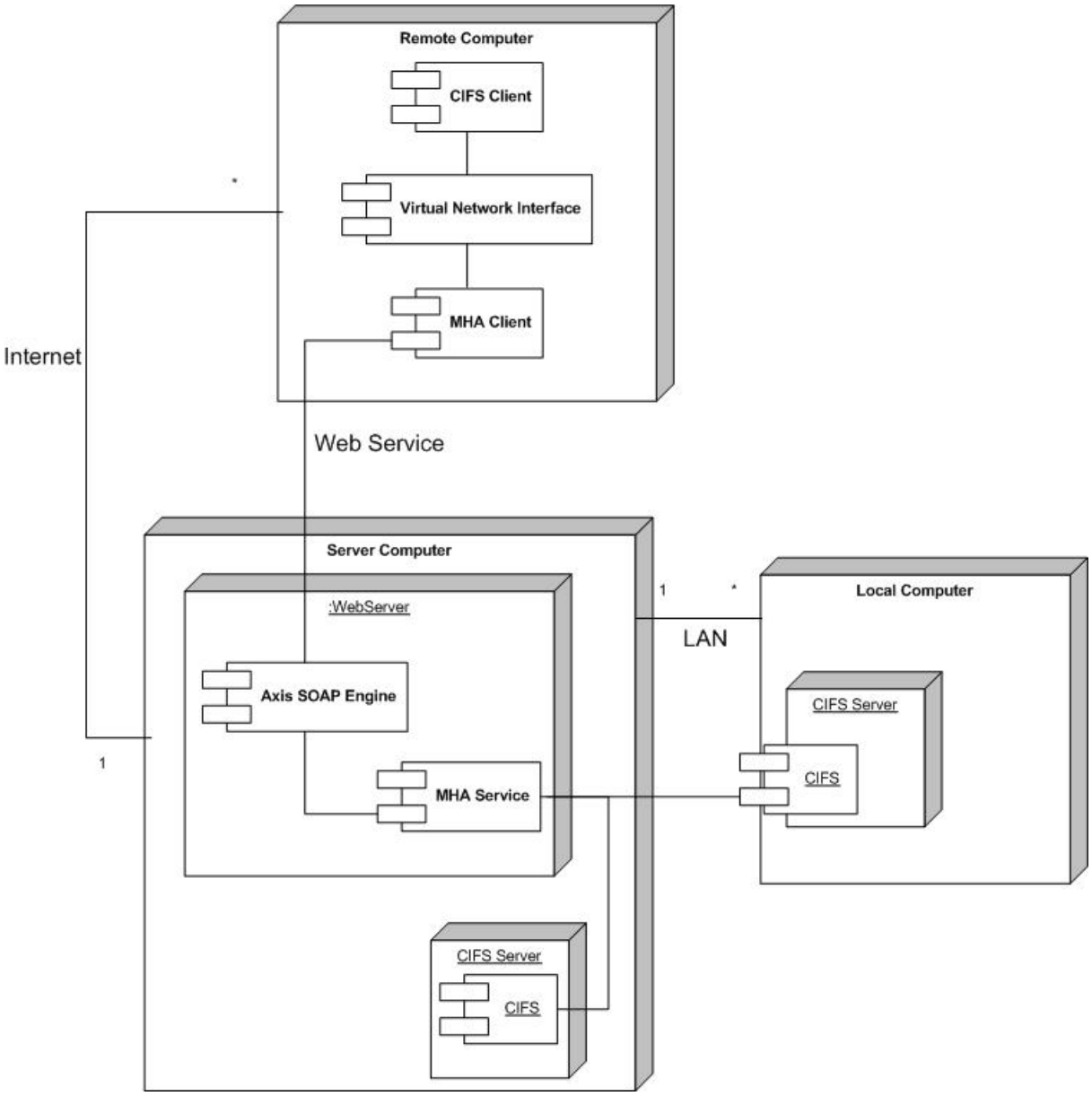

Figure 10.8 Deployment of MHA

PART III:

IMPLEMENTATION

Chapter 11 Development Environment & Tools

This chapter describes the development environment and the development tools which have been used. The decision is based on popularity, flexibility, and how well it is suited for our project. Choosing the right tools will, most of the time, increase the productivity if they are properly used. It is also crucial that those tools have a low learning curve and/or are well-documented. This helps developers to quickly grasp the techniques related for each tool instead of spending hours or days to scratch their heads to figure out how thing works.

11.1 CVS

In every software development project, there should be a way to keep track of current work and changes to the source codes. This allows several developers to work concurrently on the same files. The Concurrent Versions System (CVS) [51] is an open source implementation of such version control system.

CVS uses client-server architecture, where the server stores the current version and history of a project (including source codes and other documents) in a *repository*. The clients connect to the server to retrieve a copy of the source codes (*check-out*) and then later *commit* (also known as *check-in*) the changes to the server. Other developers perform an *update/sync* to acquire the latest changes in the repository. If the local *working copy* differs from the repository, CVS will ask the developer either to *merge* the differences or to overwrite the local changes. Thus, CVS also functions as a backup system.

The author wants to adopt the use of CVS into the thesis because of the following reasons:

- Possible to work at different places and on different computers
- A backup system; when the workstation fails, a copy of the project in the repository can be retrieved, and vice versa
- The most important reason is exactly versioning, e.g. rollback to older version if the changes cause the application to mal-function.

FYI:
The CVS server is located at www.tanvn.com and runs on Pentium 4 2.8 GHz, 1GB RAM computer with Gentoo Linux installed. Anonymous access is disabled; a user account must be created to get access.

11.2 Apache Ant

Apache Ant [52] is a Java-based build tool for automating software build processes. It is similar to *make*, the build tool for C/C++ software development in Unix. Ant is written in Java and is developed in Java software development. The primary goal of Ant is to solve the portability problem (compilation of Java source codes in different platforms) by providing a large amount of built-in functionality (i.e. tasks) which is guaranteed to behave identically on all platforms.

Unlike make and other traditional build tools, which have its own format (e.g. Makefile for make), Ant uses an XML file, *build.xml*, to describe the build process and its dependencies. The build.xml file can be configured using a property file (usually named *build.properties*), which is used to set values that change often and/or use in many places in the build file. Below is an example of a build.xml file for a typical "Hello, World!"-application. It specifies three different self-explanatory targets: *clean*, *compile* and *jar*.

```
src_dir=./src
bin_dir=./classes
jar_file=hello.jar
```

Table 11.1 Hello World example configuration, build.properties

```
<?xml version="1.0"?>
<project name="HelloWorld" default="compile">
    <property file="build.properties"/>
    <target name="clean">
        <delete dir="${bin_dir}"/>
    </target>
    <target name="compile">
        <mkdir dir="${bin_dir}"/>
        <javac srcdir="${src_dir}" destdir="${bin_dir}"/>
    </target>
    <target name="jar" depends="compile">
        <jar destfile="${jarfile}">
            <fileset dir="${bin_dir}" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class" value="HelloWorld"/>
            </manifest>
        </jar>
    </target>
</project>
```

Table 11.2 Hello World example build.xml

Ant comes with a large amount of core tasks and optional tasks. Ant is in addition flexible and lets you extend and customize your own tasks or using other tasks developed by others. This thesis uses the following Ant tasks:

- *File Tasks*
  Ant supports many file operations such as delete, move, copy, mkdir, touch, etc.
- *Archive Tasks*
  Ant can package compiled codes and resources in various formats (jar, ear, war, zip, tar)
- *Documentation Tasks*
  Many documentation tasks are supported, e.g. the *javadoc* task for generating Javadoc files
- *Testing Tasks*
  Junit and *JunitReport* tasks can be used to run tests and build test reports
- *WSDL2Java Task*
  Task for generating Java source file from WSDL

11.3 Eclipse

Eclipse [53] is a popular open source integrated development environment (IDE) which focuces on providing vendor-neutral development platform and application frameworks for building software. It provides a plug-in based framework that tool developers can easily

create, integrate and utilize the plug-ins. The Eclipse Platform is written in Java and has been deployed on a range of operating systems, including Windows and Linux.

Eclipse is developed for Java software development, but many extensions are developed which provide IDE for other programming languages (for example PHP and C/C++). Java development (in web services as Java application for Mobile Home Access) with Eclipse is provided through the Java Development Tooling (JDT), and C/C++ development (in implementation of the virtual network interface) through the C/C++ Development Tooling (CDT) [54].

Eclipse has many features which are commonly used in a modern IDEs, such as keywords highlighting, content assisting, debugging tools, etc. Other features of Eclipse of important to the thesis are:

- C/C++ development tools
- Supported on many platforms
- Ant building tools
- JUnit supported
- CVS
- Incremental compilation
- Many plug-ins available for development

11.4 Apache Tomcat

The Apache Tomcat [55] is a web container which implements the servlet and the JavaServer Pages (JSP) specification [56]. Tomcat provides an environment for Java code (web application) to run in cooperation with the web server. It uses XML-formatted configuration files that could be easily edited, or by using built-in configuration and management tools. Tomcat is written in Java, thus it could be run on nearly all platforms.

Tomcat can run as a stand-alone web server, but it is often used to provide a servlet engine to existing web server. Typically in combination with an Apache web server and *mod_jk* plug-in is used to handle the communication between the web servers. This is the reason why Tomcat runs on port number 8080 by default.

The directory structure of Tomcat, i.e. organization of files and directories in $CATALINA_HOME (the Tomcat root directory where Tomcat is installed), is listed below.

Directory	**Purpose/contains**
bin	Scripts to startup and shutdown Tomcat, and other scripts
common	For classes and resources common for web applications and internal Tomcat codes
conf	Contains configuration files. The most important is the main configuration file for Tomcat is *server.xml*.
logs	Contains access and error logs
server	Contains classes that are used internally by Catalina servlet container.
shared	For classes and resources shared across all web applications
temp	Directory used by JVM for temporary files

webapps	Automatically loaded web applications
work	Temporary working directories for web applications

Table 11.3 Tomcat file/directory structure

The primary reason for choosing Apache Tomcat prior to other web servers is that it supports Java web applications and Apache Axis.

11.4.1 Deployment Using Web Archive File

A web application must be deployed on a web container in order to be executed. For Apache Tomcat, deployment can use one of the following approaches:

- Copy unpacked directory hierarchy into a subdirectory in $CATALINA_HOME/webapps/
- Copy the web application archive file into $CATALINA_HOME/webapps/. NOTE: Updating the application requires deletion of the expanded directory that Tomcat created, replacing the web application archive file, and then restart Tomcat server.
- Use the Tomcat "Manager" web application to deploy and undeploy web applications.
- Use "Manager" Ant tasks in the build script, i.e. build.xml.
- Use the Tomcat Deployer.

We choose the second approach which uses the Web Application Archive (WAR) file format to deploy web applications. Even though, there are additional steps required using this approach but this can be solved easily by writing a (shell) script, e.g. *deploy.sh* for Unix-based systems and *deploy.bat* for Windows, to execute these operations automatically. In the latest versions of Tomcat, it is not necessary to restart the web container and the web applications are *hot deployed*. The most important reason for using this approach is to eliminate the portability problem since all web containers, which are compatible with Servlet API Specification (version 2.2 or later), are required to accept a web application archive file. Web Archive file format is the same as Java Archive (JAR) format except that it contains a web application.

Using the Apache Ant task, web services can easily be built in a WAR file that can be dropped into an application server. When the WAR file is placed into *$CATALINE_HOME/webapps/* directory, the server automatically expands the web application archive file into its unpacked form without any intervention from users.

All classes in the *WEB-INF/classes/* and those in the JAR files inside *WEB-INF/lib/* directory are visible to classes within the web application. We chose to package classes related to our application into a JAR file and place it into *WEB-INF/lib/* to avoid copying and maintaining the entire file hierarchy in *WEB-INF/classes/*. This makes the development and distribution of the web application simpler as we only need to update one single file, i.e. the JAR file.

11.5 Apache Axis

Apache Axis [57] is a SOAP engine, i.e. a framework to construct and process SOAP messages for servers, clients and gateways. It can be dropped inside a web container such as Apache Tomcat (Axis has also a "secret" mode where it can run as a stand-alone server). Axis is an up-to-date implementation of the web services specification and is able to handle upcoming specifications by W3C. This reason alone makes Axis attractive for the development of Web Services. In addition, Axis follows the rules in WS-I specifications to

maintain operability between many different implementations from the web services specification.

The configurability and flexibility in Axis makes it very simple to use. In addition, Axis contains many built-in tools that are useful for web services development. For example, the WSDL2Java and Java2WSDL code generator, and SOAP and TCP monitor.

The directory structure of Axis has the same directory layout as other web applications (Axis is in fact a web application when it is dropped inside a web container).

Files/Directory	Purpose
/	The document root of the web application is the top-level directory which is a sub-directory in $CATALINA_HOME/webapps/
/.html, /*.jsp, etc.*	Files which are visible to client browser, such as HTML pages, JSP pages, and other resources such as JavaScript, Cascading Style Sheet (CSS) and images. For small web applications, it is sufficient to place these files in application's root directory. For larger applications, it is common to organize these files into a subdirectory hierarchy.
/WEB-INF/	Contains the Web Application Deployment Descriptor, *web.xml*, which describes the servlets and other components for web application, along with any initialization parameters and security constraints. For Axis, there is in addition a *server-config.wsdd* file which contains configurations for Axis and deployed services.
/WEB-INF/classes/	This directory contains Java class files and associated resources required by the application. **Note:** If classes are organized into Java packages, this must reflect in the directory hierarchy.
/WEB-INF/lib/	This directory contains JAR files and associated resources. Usually this directory contains third party class libraries and drivers.

Table 11.4 Standard web application's directory layout

11.5.1 Service Deployment Descriptor

Deployment of services via JWS files is intended for simple services. A JWS file is actually a Java source file with the *.jws extension (instead of *.java extension) which is dropped inside Axis root directory. Axis automatically compiles JWS files, and generates WSDL file for the service. A SOAP call is then converted to Java invocation to the corresponding Java class. The deployed service using JWS is left as-is. If users want to have more control and advanced settings for web services, the web service must be deployed using **Web Service Deployment Descriptor (WSDD)**.

WSDD defines many features that can be specified for a web service. For example, we can specify the services scope which describes how a Java object responds when it receives a request. Following values are possible:

- *request* scope (default)
  A new object will be created each time a SOAP request comes in
- *session* scope
  A new object will be created for each session-enabled client
- *application* scope
  A singleton shared object is created to service all requests

A service can also specify handlers and chains to be invoked when it is called/invoked by the clients. For example we can add logging and monitoring facility.

11.5.2 WSDL2Java

WSDL2Java is a tool provided by Axis to generate Java templates for a service. WSDL2Java can be found in *org.apache.axis.wsdl* package. WSDL2Java can generate both client and/or server files.

WSDL generates the following files at the client side.

WSDL	**Java files generated**
For each entry in the type section	- A class - A holder class (if this type is used as inout/out parameter)
For each portType	- An interface
For each binding	- A stub class
For each service	- A service interface - A service implementation (the locator)

Table 11.5 Generated Java files for client side

The generated files at the server side includes

WSDL	**Java files generated**
For each binding	- An implementation template class - A skeleton class (if –skeletonDeploy parameters is set to true).
For all services	- One deploy.wsdd file - One undeploy.wsdd file

Table 11.6 Generated Java files for server side

Axis also provides Java2WSDL for the reverse process, i.e. generate WSDL document from Java source.

11.5.3 TCP Monitor & SOAP Monitor

The TCP and SOAP monitor come with Apache Axis and as the names imply, they are used to monitor the contents of SOAP packages or messages involved in communication. Developers can use these tools to verify that the services work as expected and for debugging purposes. The difference between TCP Monitor and SOAP Monitor is that TCP Monitor also displays HTTP headers (the name TCP Monitor is misleading as developers may expect TCP headers). This is useful in cases where the services are dependent of the HTTP headers.

11.6 Ethereal

Ethereal [58] is a network analyzer application (also known as packet-sniffer) which monitors data traveling over the network. This program intercepts the contents in packets at low level to get more details about protocols because it is not well-documented and to verify that it works as specified.

Network analyzer applications are commonly used in reverse-engineering to capture a protocol behaviors, for example the Samba suite that simulates and behaves as a Windows CIFS Server for Unix has used network analyzer to reveal the functions in CIFS.

11.7 Development Platform

The platform for development is Linux, using Gentoo Linux distribution [59]. The screenshot below shows Gentoo Linux with WindowMaker as the window manager and four desktop applications: Ethereal, VMWare Workstation with Windows XP Professional, Eclipse SDK, and a terminal (eterm) window.

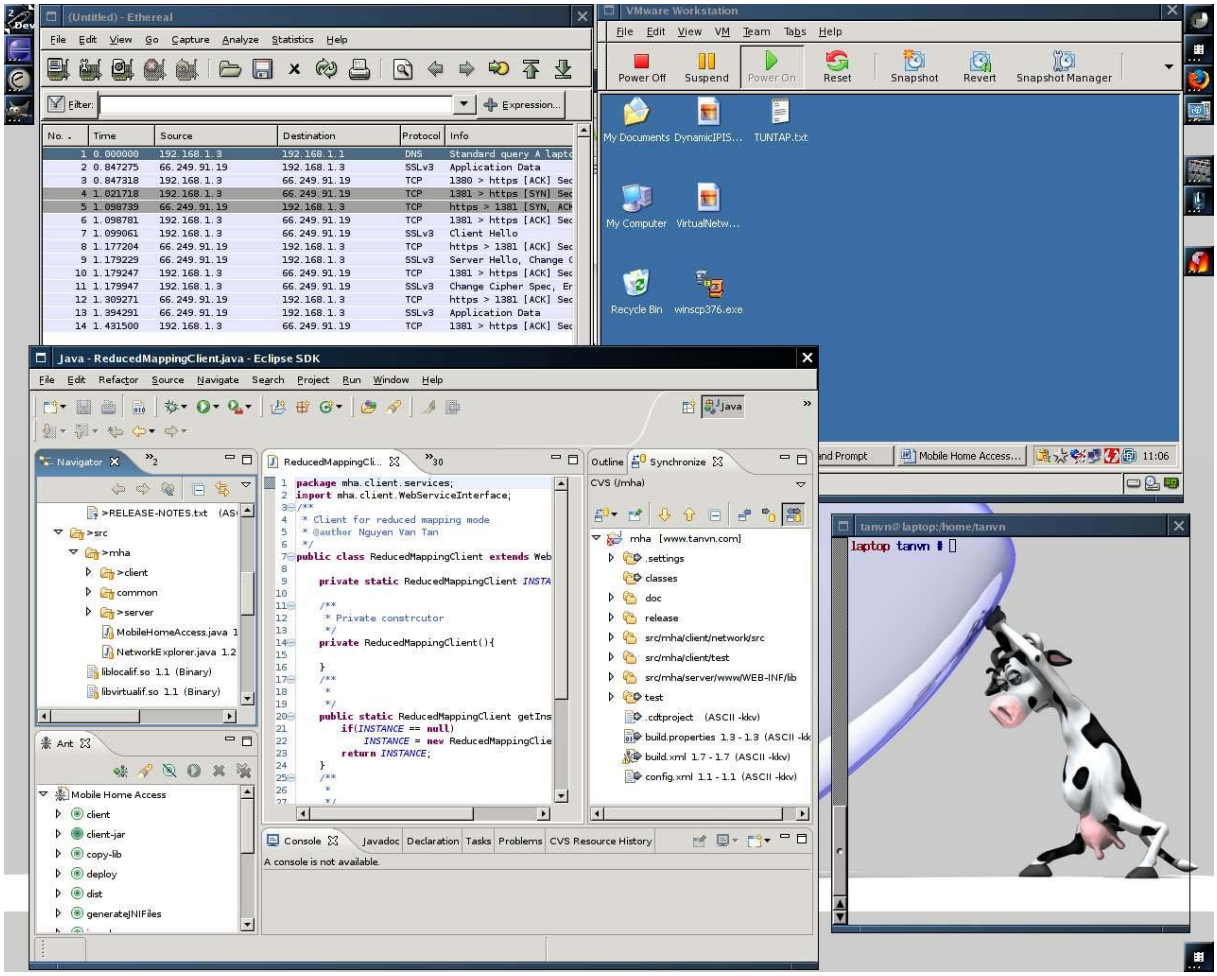

Figure 11.1 Screenshot of the development environment

Chapter 12 MHA Server

This step of the development involves implementing the web services and the interfaces clients can access. We start first to define the services using WSDL and then use the WSDL2Java tool to generate Java files. The implementation is then built based on these generated Java sources. After the implementation of the services, the solution is deployed via the use of Web Archive format to an existing Apache Tomcat installation.

12.1 Web Services

The web services for Mobile Home Access is classified into two different categories:

1. Authentication
   which consists of *login* and *logout* operations
2. Tunneling
   which consists of *put* and *get* operations

Common fault message for all operations is the *ServerOperationFault* message which the server responds with when the SOAP requests cannot be processed. Another common fault message when the clients are logged in is the SessionFault which specified that the session ID cannot be found or it has expired.

```
<wsdl:message name=”ServerOperationFault”>
  <wsdl:part name=”message” type=”xsd:string”/>
</wsdl:message>
<wsdl:message name="SessionFault">
  <wsdl:part name="message=" type="xsd:string"/>
</wsdl:message>
```

Table 12.1 Common fault messages

12.1.1 Login

Authentication is one important service that restricts access for unauthorized users. Generally the home services should only be available for users at home network, and for them only. The procedure for authentication involves two operations, login and logout.

For every session, the clients must be identified by logging in with a username and password. The server then responds with a session ID if the user is successfully authenticated or sends an AuthenticationFault message to notify that the user could not be identified using the supplied username and password.

```
<!-- Message -->
<wsdl:message name=”loginRequest”>
   <wsdl:part name=”username” type=”xsd:string”/>
   <wsdl:part name=”password” type=”xsd:string”/>
<wsdl:message>
<wsdl:message name=”loginResponse”>
   <wsdl:part name=”sessionID” type=”xsd:string”/>
</wsdl:message>
<wsdl:message name=”AuthenticationFault”>
  <wsdl:part name=”message” type=”xsd:string”/>
</wsdl:message>
<!-- Interface -->
<wsdl:portType ...>
```

```
  <wsdl:operation name=”login”>
   <wsdl:input message=”tns:loginRequest”/>
   <wsdl:output message=”tns:loginResponse”/>
   <wsdl:fault name=”AuthenticationException”
               message=”tns:AuthenticationFault”/>
   <wsdl:fault name=”ServerOperationFault”
               message=”tns:ServerOperationFault”/>
  </wsdl:operation>
</wsdl:portType>
<!-- Binding -->
<wsdl:binding ...>
   <wsdl:operation name=”login”>
      <wsdl:input/>
      <wsdl:output/>
      <wsdl:fault name=”AuthenticationFault”>
          <soap:fault name=”tns:AuthenticationFault” use=”literal”/>
      </wsdl:fault>
      <wsdl:fault name=”ServerOperationFault”>
          <soap:fault name=”tns:ServerOperationFault” use=”literal”/>
      </wsdl:fault>
  </wsdl:operation>
</wsdl:binding>
```

Table 12.2 Login WSDL definition

12.1.2 Logout

Logout is a one way operation from the client. When the server receives a logout request, the server closes all connections, session and other resources associated to the session ID.

```
<!-- Message -->
<wsdl:message name=”logoutRequest”>
   <wsdl:part name=”sessionID” type=”xsd:string”/>
</wsdl:message>
<!-- Interface -->
<wsdl:portType ...>
  <wsdl:operation name=”logout”>
     <wsdl:input message=”tns:logoutRequest”/>
  </wsdl:operation>
</wsdl:portType>
<!-- Binding -->
<wsdl:binding ...>
  <wsdl:operation name=”logout”>
     <wsdl:input />
  </wsdl:operation>
</wsdl:binding>
```

Table 12.3 Logout WSDL definition

12.1.3 Put

The Mobile Home Access clients send CIFS packets using the put operation. The message contains CIFS request(s) attached and also the session ID which verifies that the client is authenticated. The session ID is also used to identify objects associated to the client at the server-side. Even though the HTTP session can be used with Apache Axis, the internal session handling is preferred, because this enables a client to create many (HTTP/TCP) connections to the same Tunneling service.

```
<!-- Message -->
<wsdl:message name="sendCIFSRequest">
  <wsdl:part name="sessionID" type="xsd:string"/>
```

```
  <wsdl:part name="CIFSRequest" type="xsd:base64Binary"/>
</wsdl:message>
<!-- portType -->
<wsdl:portType ...>
 <wsdl:operation name="put">
  <wsdl:input message="tns:sendCIFSRequest"/>
  <wsdl:output message="tns:empty" />
  <wsdl:fault name="SessionFault" message="tns:SessionFault"/>
  <wsdl:fault name="ServerOperationFault"
message="tns:ServerOperationFault"/>
 </wsdl:operation>
</wsdl:portType>
<!-- Binding -->
<wsdl:binding ...>
 <wsdl:operation name="put">
  <wsdl:input>
    <mime:multipartRelated>
      <mime:part>
         <soap:body use="literal" parts="sessionID"/>
      </mime:part>
      <mime:part>
        <mime:content part="CIFSRequest" type="multipart/mixed"
use="literal"/>
      </mime:part>
    </mime:multipartRelated>
  </wsdl:input>
  <wsdl:output />
    <wsdl:fault name="SessionFault">
      <soap:fault name="tns:SessionFault" use="literal" />
    </wsdl:fault>
    <wsdl:fault name="ServerOperationFault">
      <soap:fault name="tns:ServerOperationFault" use="literal"/>
    </wsdl:fault>
 </wsdl:operation>
</wsdl:binding>
```

Table 12.4 Put WSDL definition

When the server receives a put request, the server first verifies the session ID and responds a SessionFault message if the session ID could not be found by the session manager. If the session is valid, the packets are retrieved and put into the queue for processing. The server is responsible for forwarding these packets in the queue to the right destination in the home network. This involves creating an internal CIFS client (i.e. a TCP connection) and sends those requests to the CIFS server. The routing information is included for each packet in the attachment (we'll see this clearly in the SOAP message sent from the clients in the next chapter).

12.1.4 Get

In figure 10.4, the server sends a response message which includes CIFS responses to clients. The Axis implementation does not provide a notification message, i.e. one-way operation message from the server to clients. Thus the get operation is implemented by using a request-response sequence.

```
<!-- Message -->
<wsdl:message name="getCIFSResponse">
  <wsdl:part name="CIFSResponse" type="xsd:base64Binary" />
</wsdl:message>
<wsdl:message name="pullCIFSResponse">
```

```
  <wsdl:part name="sessionID" type="xsd:string" />
</wsdl:message>
<!-- portType -->
<wsdl:portType ..>
 <wsdl:operation name="get">
   <wsdl:input message="tns:pullCIFSResponse" />
   <wsdl:output message="tns:getCIFSResponse"/>
   <wsdl:fault name="SessionFault" message="tns:SessionFault"/>
   <wsdl:fault name="ServerOperationFault"
message="tns:ServerOperationFault"/>
 </wsdl:operation>
</wsdl:portType>
<-- binding -->
<wsdl:binding ...>
 <wsdl:operation name="get">
    <wsdl:input />
    <wsdl:output>
      <mime:multipartRelated>
        <mime:part>
          <mime:content part="CIFSResponse" type="multipart/mixed"
use="literal" />
        </mime:part>
       </mime:multipartRelated>
     </wsdl:output>
     <wsdl:fault name="SessionFault">
        <soap:fault name="tns:SessionFault" use="literal" />
     </wsdl:fault>
     <wsdl:fault name="ServerOperationFault">
        <soap:fault name="tns:ServerOperationFault" use="literal" />
     </wsdl:fault>
 </wsdl:operation>
</wsdl:binding>
```

Table 12.5 Get WSDL definition

12.2 Server IP Address

Remote clients must specify an IP address or a domain name address to locate the Mobile Home Access server (i.e. address to the home network). The address must be unique on the Internet, also known as public or global IP addresses. When users connect to the Internet, for example via a broadband technology such as ADSL (Asymmetric Digital Subsriber Line), an IP address is allocated to the network device. On the home network, a router is connected between the ADSL modem and computers in the network. The router has two IP addresses associated. One IP address for the outbound network, i.e. for the Internet, and the other for home network. It is important that the outbound IP address is a public IP address. Otherwise, home users can purchase one or more public IP addresses from the ISP for some extra fees.

There are two ways IP address is allocated, either statically or dynamically:

1. **Static**
   The ISP can assign a static IP address to its customers (the IP address can map to customers' MAC[10] address or using other approaches). When the IP address is static, there is no need to do anything as the home network can always be accessed via the same IP address. Today, domain name service providers offer a domain name at a low cost to map a user-friendly name to the home network's IP address. This domain name is easier to remember than an IP address.

[10] Media Access Control (MAC) is a globally unique address of the network interface. The address is normally burned-in by the manufacture.

2. **Dynamic**
   ISPs have limited available IP addresses, and therefore use dynamic IP allocation to assign addresses from a small pool to a larger number of customers. The Dynamic Host Configuration Protocol (DHCP) is used to dynamically assign addresses. It is commonly used in dial-up access via analog modem, where new IP address is assigned for each dial-up connection. For users with broadband access, there is a lease time in DHCP which determines how long a client can use an address before requesting its renewal. This mechanism allows addresses to be reclaimed if a client goes offline. When a client reconnects, it is not guaranteed that it will be assigned the same address.

It is obvious that we need a service for discovering the IP address of home network. The service could be implemented in:

1. Internet Service Provider
2. Third-party

Note:
In the text and diagrams below, the author says intentionally that Mobile Home Access clients send requests to the (DNS) server to resolve hostname to IP address of the home network. This is incorrect as this is handled internally by DNS client in the operating system. Home users only type in the hostname which then triggers such requests to be sent if there is no such entry in the cache.

12.2.1 Internet Service Provider

Internet Service Providers are potential providers for the service as they already run a DHCP server to allocate customers' IP addresses. The solution proposes that Internet Service Provider maintains its IP addresses which are in used and to which customer. For example, when a customer connects to the Internet, for ADSL using PPPoE the username and IP address that assigned to the customer is stored. The ISP can create a domain name associate for each customer (for example for Telenor Online, *customername.online.no* could be the domain name for a username *customername*).

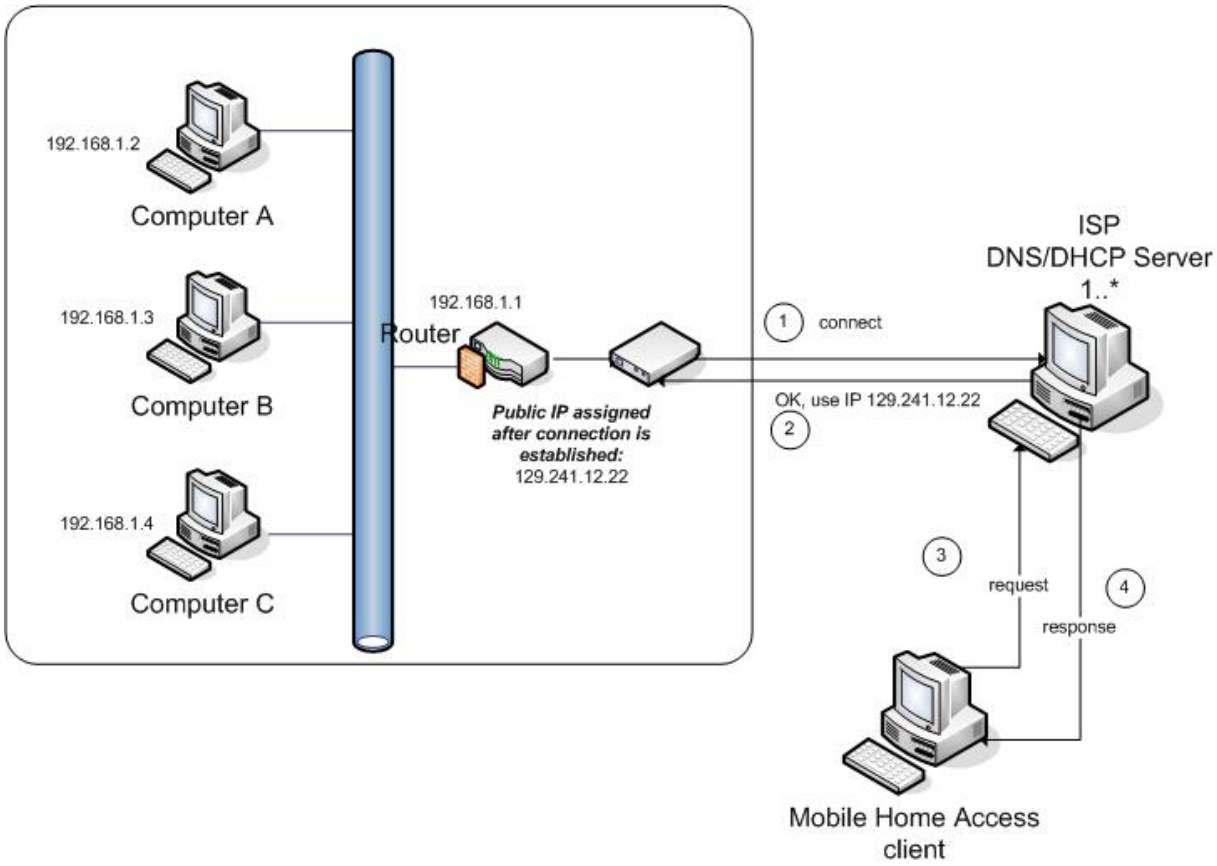

Figure 12.1 ISP Dynamic IP allocation and resolving

The domain name for a home network is static, but the IP address can change over time. When the DHCP server assigns an IP address to a client, this address is updated. The current IP address for a home network can be resolved from the domain name by sending a request to ISP's DNS server. By using this approach, a home network is identified by a name which is easier for a customer to remember.

12.2.2 Third-party

This solution puts the discovery service in one or more servers running "outside" (i.e. elsewhere in the Internet) the home networks. The server maintains a database which could contain username and the current IP address of a home network. A small client application which is running on one of the computers in the home network that notify the current IP address to the server. The notification message could be done periodically or whenever it discovers that the public IP address has been changed (see point 1 in figure below).

When the Mobile Home Access clients want to connect to the home network, the user's domain name (for example *username.3rdparty.com*) is resolved by sending a request to the server which returns a response IP address of the home network and thus the Mobile Home Access server (see point 2 and 3 in the figure below).

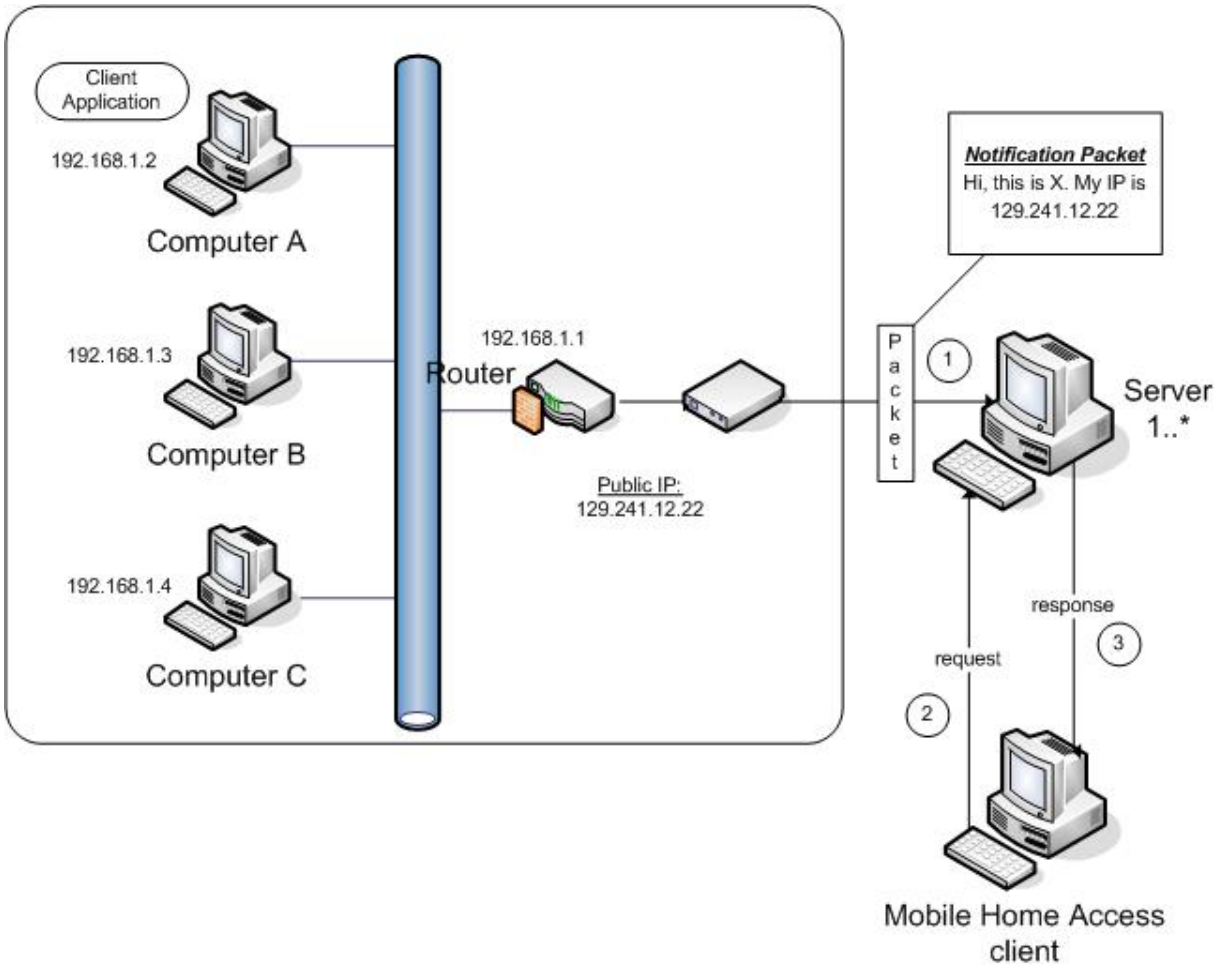

Figure 12.2 Resolve Server IP Address using a third-party DNS

There are many free dynamic DNS (Domain Name Service) providers, that offer such functionality. Examples of free DNS service provider for dynamic IP address registration are No-IP (www.no-ip.com) and DynDNS (www.dyndns.org).

12.3 Homepage

A website for Mobile Home Access can be setup at the home network to indicate that Mobile Home Access server is successfully installed (by accessing the website in a web browser) similar to the installation of Apache Axis. In addition, users can view a list of deployed services of their installation. A general website for Mobile Home Access can also be created to serve as a information resource for Mobile Home Access users.

Examples of usage:

- Download the sources of the application
- Java APIs, user guides and other documentation could be available
- A forum could be setup to enable users to get an answer for their problems from other users and/or developers.

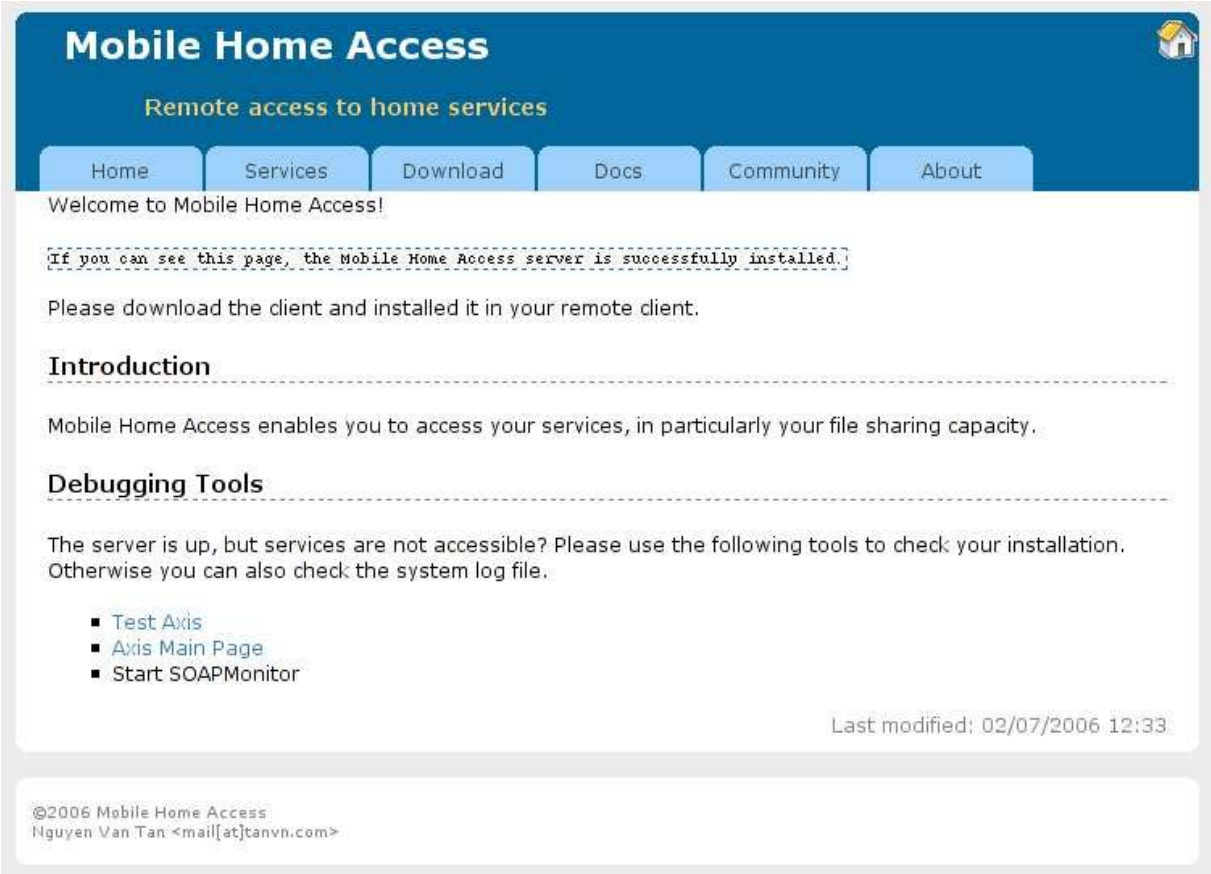

Figure 12.3 Mobile Home Access Homepage

Note:
This website (available at http://mha.tanvn.com) is created for example purpose only. The content is poor to be considered as productive.

12.3.1 Mobile Home Access File Structure

The file structure for Mobile Home Access web application has the same directory layout as Axis (it is actually extended from the Apache Axis directory layout) with these additions:

Files/Directory	Purpose
/bin/	Contains scripts that ease the development and deployment process
/dist/	Distribution directory in which binary sources for clients and/or server application can be downloaded.
/help/	Contains documentations for installation guides, howto guides, etc.
/images/	Images used by the web application
/javadoc/	Contains Java APIs for Mobile Home Access
/javascript/	Javacript files
/styles/	Contains Cascading Stylesheet (CSS) files
/testreports/	Contains test reports

Table 12.6 Mobile Home Access directory layout

Chapter 13 MHA Client

Exposing the services at the home network is done via the Mobile Home Access server which sends requests on behalf of clients and sends responses to clients. The challenge is to capture CIFS requests on client's computer and forward them to the Mobile Home Access client so it can be manipulated and sent as attachment to the SOAP message. Unfortunately, there exists no simple and pure solution that enables a CIFS client to forward packets to an application instead of putting them on the wire. A common solution to this problem is to use a virtual network interface, which has been used in many VPN clients (e.g. OpenVPN, Cisco VPN client, and vpnc).

This thesis will use the same approach and implement a virtual network interface (TUN/TAP device) on the clients' side to intercept CIFS requests and send them to Mobile Home Access client for further processing, i.e. encapsulate in SOAP message as attachement. The development is targeted for Linux platform in the initial phase, and testing for the Windows platform will be performed later. The Linux kernel is compiled with the built-in support for TUN/TAP device [60], but can also compile as a module which can be loaded at runtime.

13.1 Web Services

The Mobile Home Access clients communicate with the Mobile Home Access server using web services. The clients construct SOAP messages and then send over a HTTP connection. The interaction using web services work similarly as method calls in Java. Apache Axis encapsulates and decapsulates all SOAP messages to and from the Mobile Home Access application. The SOAP Monitor is used to see the actual messages put on the wire.

13.1.1 Login

The application provides a GUI to let a user enter the login information, i.e. username and password. The MHA client then constructs a login request and sends it to the server.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
     xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
     xmlns:xsd=http://www.w3.org/2001/XMLSchema
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
     <login xmlns="">
       <username xsi:type="xsd:string">tanvn</username>
       <password xsi:type="xsd:string">12345</password>
     </login>
  </soapenv:Body>
</soapenv:Envelope>
```

Table 13.1 Login request example

If the authentication failed, because the user has typed wrong username and/or password, the server responds with a SOAP fault message. The application displays an error message and lets the user to re-enter the correct username and password.

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope
```

```
     xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
     xmlns:xsd=http://www.w3.org/2001/XMLSchema
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <soapenv:Fault>
       <faultcode>soapenv:Server.generalException</faultcode>
       <faultstring></faultstring>
       <detail>
         <message>Wrong username and/or password</message>
         <ns1:exceptionName xmlns:ns1="http://xml.apache.org/axis/">
            mha.server.services.TunnelingMode.AuthenticationFault
         </ns1:exceptionName>
         <ns2:hostname xmlns:ns2="http://xml.apache.org/axis/">
            127.0.0.1
         </ns2:hostname>
       </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Table 13.2 Authentication failed

When the user is successfully authenticated, the server responds with a session ID in the login response message.

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope
     xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
     xmlns:xsd=http://www.w3.org/2001/XMLSchema
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
     <loginResponse xmlns="">
       <sessionID>XXXXX</sessionID>
     </loginResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Table 13.3 Login response – successfully authentication

13.1.2 Logout

An example of a logout request sent from a client:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
     xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
     xmlns:xsd=http://www.w3.org/2001/XMLSchema
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <logout
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <sessionID xsi:type="xsd:string">XXXXX</sessionID>
    </logout>
  </soapenv:Body>
</soapenv:Envelope
```

Table 13.4 Logout request example

13.1.3 Put

The Mobile Home Access client is responsible to intercept packets from CIFS clients and then send them to the Mobile Home Access server.

An example of a put SOAP request including HTTP headers:

```
POST /mha/services/TunnelingService HTTP/1.0
Content-Type: multipart/related;
type="text/xml";start="<27EB27A75992CDBC79899BAEC7409715>";
      boundary="----=_Part_1_32459563.1150823604695"
Accept: application/soap+xml, application/dime, multipart/related, text/*

User-Agent: Axis/1.4
Host: localhost:8888
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 1034
Cookie: JSESSIONID=44323327F5B80756E733F15CF3CE12D4

------=_Part_1_32459563.1150823604695
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: binary
Content-Id: <27EB27A75992CDBC79899BAEC7409715>
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
     xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
     xmlns:xsd=http://www.w3.org/2001/XMLSchema
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <put xmlns="">
      <sessionID>XXXX</sessionID>
      <CIFSRequest href="cid:7D60CBCECF480DA4A8E20A7F60EF8554"/>
    </put>
  </soapenv:Body>
</soapenv:Envelope>

------=_Part_1_32459563.1150823604695
Content-Type: multipart/mixed; boundary="--=_Part_0_6296823.1150823604605"
Content-Transfer-Encoding: binary
Content-Id: <7D60CBCECF480DA4A8E20A7F60EF8554>

------=_Part_0_6296823.1150823604605
Content-Type: application/octet-stream
DstHost: 192.168.100.112
DstPort: 445
SrcPort: 34567
(binary contents…)
------=_Part_0_6296823.1150823604605
--------=_Part_1_32459563.1150823604695--
```

Table 13.5 Put Request including HTTP header

Note that the attachment is referenced by the Content-ID in the SOAP message and the “Content-Type: multipart/related” HTTP header to identicate the SOAP package. The *multipart/mixed* MIME header indicates that there can exist multiple packets inside the attachment. Each packet has related routing headers, which are extended by the Mobile Home Access client. These headers (destination host, destination port, and source port) are important for Mobile Home Access to be able to forward these packets to the right destination at home network and then send responses to the right CIFS client on the remote computer.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
     xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
     xmlns:xsd=http://www.w3.org/2001/XMLSchema
```

```
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
      <putResponse xmlns=""/>
  </soapenv:Body>
</soapenv:Envelope>
```

Table 13.6 Put Response

The put response message from the server indicates that the SOAP message is received and is successfully processed.

13.1.4 Get

A client sends a get request periodically to check/retrieve any CIFS responses from the server.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
     xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
     xmlns:xsd=http://www.w3.org/2001/XMLSchema
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
   <get soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <sessionID xsi:type="xsd:string">XXXXX</sessionID>
   </get>
 </soapenv:Body>
</soapenv:Envelope>
```

Table 13.7 Get Request example

If there are no response packets from CIFS servers, the Mobile Home Access server sends a null message to client to indicate this.

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope
     xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
     xmlns:xsd=http://www.w3.org/2001/XMLSchema
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
   <getResponse xmlns="">
      <CIFSResponse xsi:nil="true"/>
   </getResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Table 13.8 Get Response – null

When the Mobile Home Access server has (response) packets from CIFS servers the client is requesting, it will send a response message with packets in the attachment.

```
HTTP/1.1 200 OK
Content-Type: multipart/related;
type="text/xml";start="<27EB27A75992CDBC79899BAEC7409715>";
      boundary="----=_Part_1_32459563.1150823604695"
Accept: application/soap+xml, application/dime, multipart/related, text/*

User-Agent: Axis/1.4
Host: localhost:8888
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 1034
```

```
Cookie: JSESSIONID=44323327F5B80756E733F15CF3CE12D4

------=_Part_1_32459563.1150823604695
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: binary
Content-Id: <27EB27A75992CDBC79899BAEC7409715>
   <?xml version="1.0" encoding="UTF-8"?>
      <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <soapenv:Body>
            <get xmlns="">
               <CIFSRequest href="cid:7D60CBCECF480DA4A8E20A7F60EF8554"/>
            </get>
         </soapenv:Body>
      </soapenv:Envelope>

------=_Part_1_32459563.1150823604695
Content-Type: multipart/mixed; boundary="--=_Part_0_6296823.1150823604605"
Content-Transfer-Encoding: binary
Content-Id: <7D60CBCECF480DA4A8E20A7F60EF8554>

------=_Part_0_6296823.1150823604605
Content-Type: application/octet-stream
DstHost: 192.168.100.112
DstPort: 445
SrcPort: 34567
(binary contents…)
------=_Part_0_6296823.1150823604605
--------=_Part_1_32459563.1150823604695--
```

Table 13.9 Get response example

The destination host and port headers related to each packet refers now to the CIFS server which the response is coming from. The Mobile Home Access client will then send the packet to the CIFS client on the TCP port specified in the source port header.

13.2 Virtual Network Interface

The virtual network interface enables the user space application to read from received buffer instead of sending them to the kernel and to a physical network interface. On the other side, packets written to the virtual network interface from the kernel will be passed to the application.

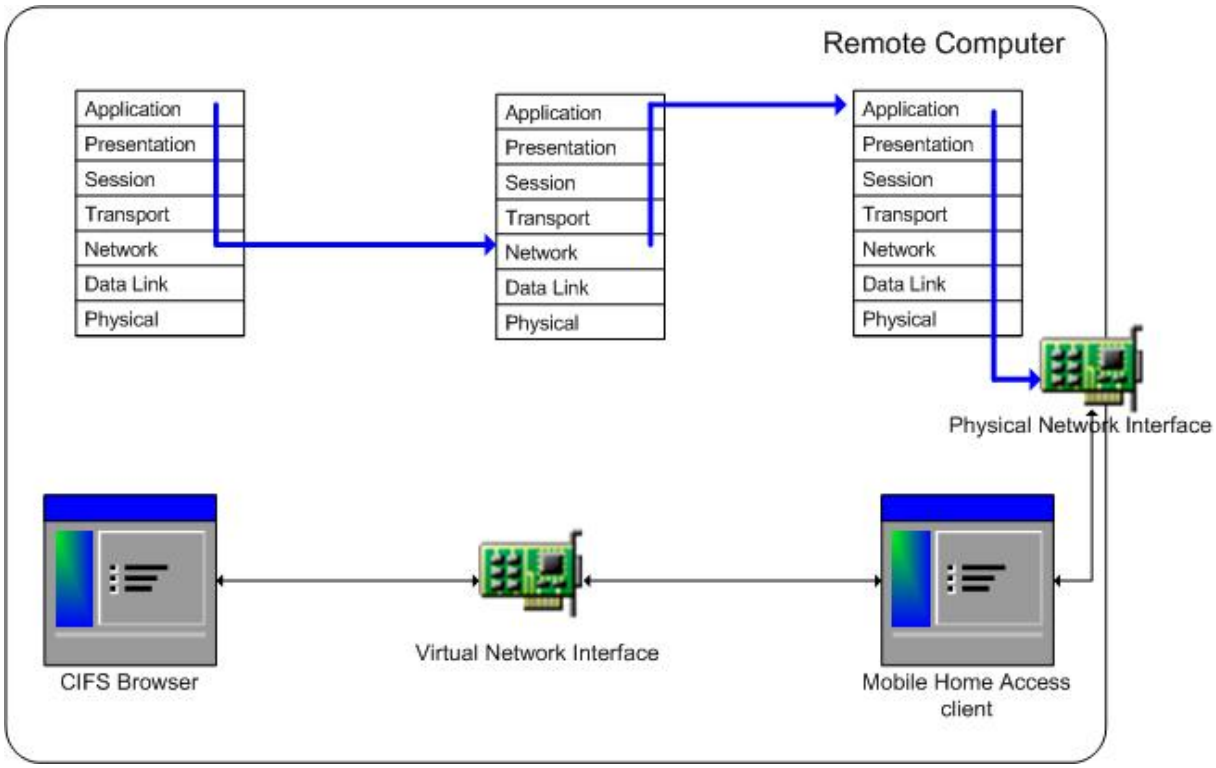

Figure 13.1 Virtual Network Interface

The TUN/TAP device is an implementation of a virtual network device/interface, developed by Maxim Krasnyansky and Maksim Yevmenkin as part of the VTun project. The TUN/TAP device was developed for Linux, FreeBSD and Solaris, but has now been ported to many platforms, including MAC OS, OpenBSD, and importantly also for Windows.

13.2.1 TUN versus TAP

The generic TUN/TAP has two modes:

- TUN
  provides a virtual Point-to-Point network device that supports IP tunneling
- TAP
  provides a virtual Ethernet network device that supports Ethernet tunneling

In TUN mode, the virtual device handles IP packets as opposed to TAP mode which handles Ethernet frames. Which mode will be used depends on the level of abstraction we want the application to handle. In the specified system, CIFS packets are exchanged between the server and clients. This means that TUN mode is sufficently for our application to work. Unfortunately, only TAP mode is supported for Windows via Win32-Tap Device (as in OpenVPN). Consequently, choosing TAP mode for all implementations makes porting to other operating systems easier.

In TAP mode, the application has more control of the packets because the application must construct and process the low level Ethernet frames, but it also has many implications. For Mobile Home Access, where the CIFS protocol communicates over TCP and UDP, the network interface must handle all headers in IP, TCP, UDP protocols. Thus, it is complicated to implement it properly because we must strip the headers in the packet and construct our own protocol headers to enable CIFS protocol to work over Mobile Home Access's web services.

13.2.2 Linux kernel configuration

TUN/TAP device requires support in the kernel, this can be easily accomplished by recompiling the kernel if it's not already supported.

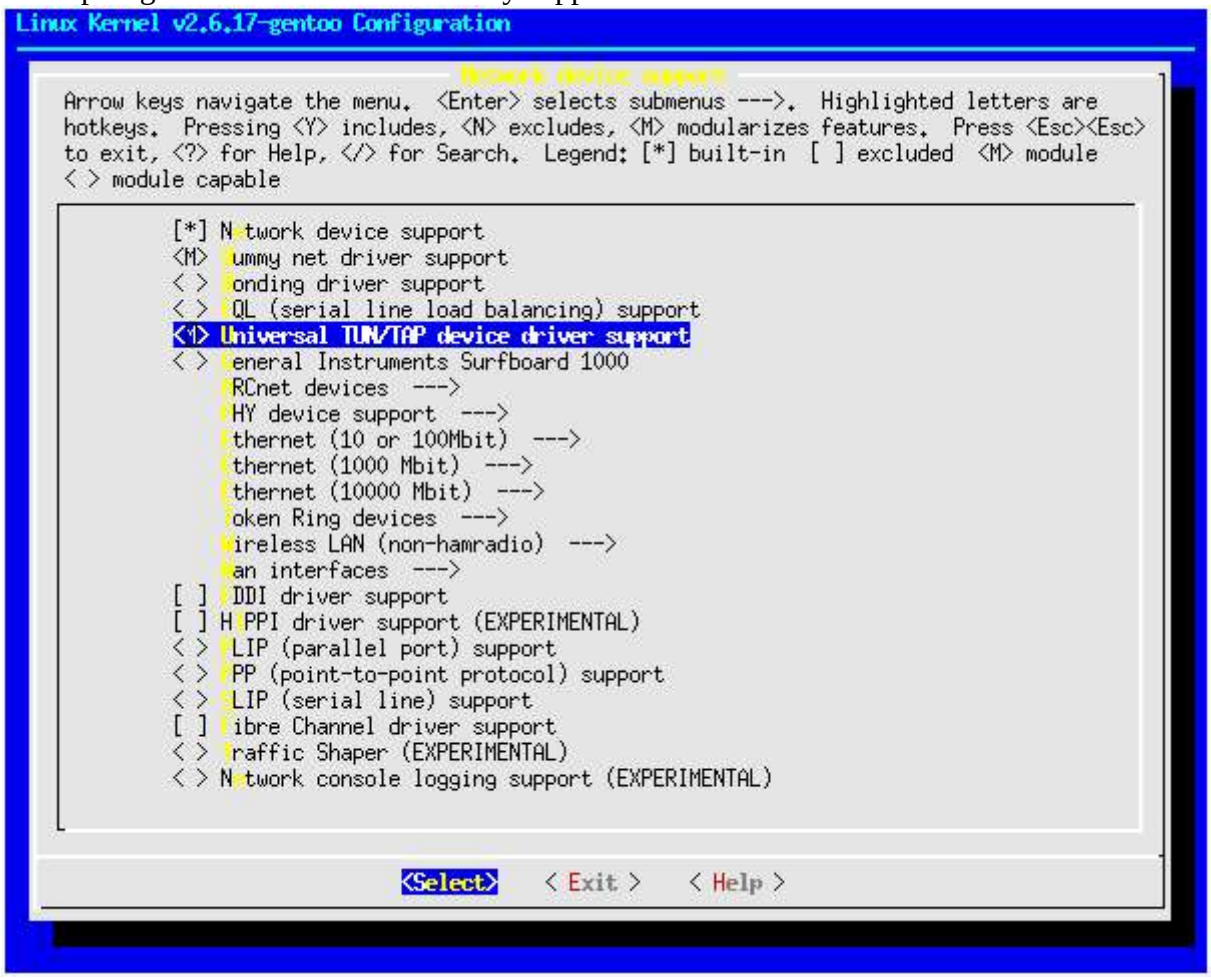

Figure 13.2 Linux kernel configuration

cd /usr/src/linux *# Linux kernel configuration* *make menuconfig* *# Compiling the kernel and update LILO Boot Manager* *make && make modules_install && dolilo* *# Loading TUN/TAP module* *modprobe tun*

Table 13.10 Edit kernel config and recompile the kernel

To build the TUN/TAP module, fire up the Linux kernel configuration and then navigate to *Device Drivers* → *Network device Support* → *Universal TUN/TAP device driver support* and mark it (**M** to build as module or ***** to compile as built-in). In figure 13.2, the TUN/TAP device is compiled as a module. Recompile the kernel and update the boot manager. Restart the computer and then load the module (alternatively, you can add the module's name to the kernel's autoloading configuration file - /etc/modules.autoload.d/kernel-2.X, where X is the kernel's minor version).

13.2.3 Routing to Virtual NIC

Remote clients must be able to specify a computer in the home network, either by a (computer-) name or IP address. We must implement addressing scheme so the Mobile Home Access server can route it to the right destination.

IP Addresses	Often used in...
192.168.1.1 – 192.168.255.254	private home networks
10.0.0.1 – 10.255.255.254	private networks in large companies and organizations

Table 13.11 Commonly used private IP addresses

The IP address to the virtual network interface is chosen to be 172.20.20.20, which is unlikely to conflict with other systems and private networks.

13.2.4 Intercommunicate with Java

When a packet is received at the virtual network interface, i.e. at the native application, it will be forwarded to the Mobile Home Access client application via the Java Native Interface. The Java application should support the following common functions for the virtual network interface.

Functions	Purpose
socket	Initiate the socket
open	Open the virtual network interface
close	Close the virtual network interface
sendto	Send packets to virtual network interface
recvfrom	Receive packets from virtual network interface

Table 13.12 Virtual Network Interface functions

Chapter 14 Experiences

This chapter lists the author's personal experiences as well as pitfalls during the development process. The purpose is to help others developers, which will continue the development of Mobile Home Access, to avoid doing the same mistakes or taking another "better" approach if appropiate. The author has chosen the "*hard right than easy wrong*" approach to look up information as well as in implementation.

Vast amounts of Technologies
The most difficult about this thesis is the involvement of many different technologies. Even though the research project done earlier has helped quite a lot, there are still many unclear issues about the protocols and how they work in practice.

Bad Documentation
When a protocol is widely used, but lacks documentation and standardization documents it is difficult to know what is right and what is wrong. The information found in the Internet may also be outdated. This relates to the CIFS protocol, and the TUN/TAP device.

Immature Technology
The Web Service Architecture is still a new born technology and many specifications and recommendations are proposed to guide and introduce the developer. Unfortunately, many of those documents do not dictate how things should be understood or implemented, and are therefore open for interpretations. The result is many different implementations may exist from the same specifications. The author was confused because there are too many different approaches to the same solution.

Design and Implementation Gap
Going from the system's design to the implementation is, in the author's opinion, a big step. The gap leaves spaces for many different interpretations and mistakes may occur. The design must be verified and reviewed carefully before the implementation of the system can take place.

Backup and Restore
Nothing is worse than losing all the data without any clue for recovery. The author has stepped on the same footstep way too many times, but still has not learned the expensive lesson to regularly backup the data. Days of work has gone into the black hole because of what you may think never happen actually happens. That is the reason why the author has setup a CVS repository, just in case one of the systems fails, the data can be recovered. Data should be checked in or backup/saved as often as possible, to avoid the moment of frustration.

Technical Resources
The author has been using open source community, such as Newsgroups, Internet Relay Chat (IRC), and forums, as a tool for finding and requesting the information needed. Technical articles from IBM, Sun Microsystem, and Microsoft are consulted as well as RFCs and Recommendations. The author has carefully verified the sources of the articles, and considered those from large software companies as reliable (such as IBM, Sun, Microsoft). Articles written by unknown persons (e.g. from personal blogs or homepage) are considered

as informative only and must be verified against the technical specifications before they can be used.

PART IV:

EVALUATION

Chapter 15 Test

Testing is a major step in development process. As the codes mature, the maintenance is hard when developers must manually write code for every test case in the application source itself. Semantically errors are easy to detect but logical errors is difficult to discover. It is very important to run test for verifying the system before releasing the final product.

Testing is rarely practiced or it is often performed in an unsystematic and error-prone way in today's software development projects. Testing should be systematic and based on common techniques and methods for verifying the system. The main purposes of testing are:

- To verify the interaction between objects
- To verify the proper integration of all components of the software
- To verify that all requirements have been correctly implemented
- To identify and ensure defects are addressed prior to the deployment of the software

15.1 JUnit

JUnit [61] is a unit test framework for Java. Unit test validates that a module or unit of the source code is working properly. Each test case is written in a separate Java source file. A test case can construct *mock objects*, which mimic the behaviors of real objects, to test the behavior of other objects. When changes occur to the source codes, the test cases help to quickly identify and fix problems that are caused by regression.

JUnit framework makes writing test an easy task for developers. JUnit automates the testing, and can be scheduled to run at anytime of the day. Another advantage of using JUnit is the documentation value of the test cases, e.g. developers can look at the unit tests to see how to use the module, in addition to gain a basic understanding of the API.

```
public class TestCalculation extends TestCase{
  public void testMultiplication(){
      assertEquals("Multiplication", 4, 2*2);
  }
}
```

Table 15.1 Test example for multiplication

15.1.1 Test Report

This thesis has performed unit tests on core functionalities of Mobile Home Access's Tunneling Mode. This includes tests on:

- Web Services operations of Mobile Home Access (e.g. put and get)
- CIFS operation with remote CIFS clients

Because the implementation of virtual network interface is incomplete, it is not possible to perform full system tests. The tests of CIFS operation are done by first capturing the "local" CIFS packets using Ethereal network analyzer. Those packets are then sent to the Mobile Home Access server, and the responses from the server are verified using the captured packets.

Test reports of unit tests are generated by Ant task and available at http://mha.tanvn.com/ under Documentation section. Tests on graphical user interfaces have not been performed or written. It is recommended that these test cases to be written in future development as the end users normally.

Chapter 16 Performance

The performance of a system is a measurement of how quickly the system responds to an action initiated by the system or by the user. Performance testing is performed under a particular workload to test how well the system handles it. Nowadays, system performance is as important as other aspect of the system. This is because the computer power has increased as well as the hardware components are getting cheaper and cheaper. Thus, the users expect that the response is faster. For Mobile Home Access, the performance tests will be compared to the local CIFS traffic on the local area network and then which improvements can be done to optimize the throughput and performance of the system.

16.1 Bottlenecks

It is a good idea to figure out (from the architecture view or by noticing the system's behaviours) the bottlenecks of the system before we start, and then perform some test in part of the system to check if it cause the system to perform badly.

16.1.1 Internet Bandwidth

Mobile Home Access provides CIFS services over Internet connection opposed to regular CIFS services which operate in the local area network. SOAP messages with CIFS packets as payload traverse through the Internet before they reach the final destination at the other end of the Internet (either at home network or at remote networks which could be anywhere). Thus, the Internet bandwidth is of great important to the response time.

Unfortunately, the actual speed/bandwidth of an Internet connection is dependent by many factors. The most important is the download and upload speed provided by the broadband operators. Though the customers have the same bandwidth, the actual speed may vary as the speed decreases when the distance to the central server increases. In addition, the time of the day when there is high Internet traffic will also decrease your Internet speed. For example, you may experience higher speed at night than during daylight. When we connect to remote servers, the response time is shorter when the server is nearby than for others located far from the clients.

The bandwidth/speed of an Internet connection is thus undetermined and must be taken into account when we measure other performance metrics (such as latency, round-trip, response time) related to the Internet connection speed. It is necessary to use a large sample and calculate the average value of them to be more accurate.

16.1.2 Processing time

Processing time is the time from a job/request is received till a response is put on the wire. For Mobile Home Access, this is the time a CIFS request from the local CIFS client is sent to Mobile Home Access clients till a SOAP message is constructed and put on the wire. This is highly undetermined as it is dependent on the hardware installed on the specific system and how busy the system is (which results that the waiting time increases till the request is processed).

16.1.3 Protocol Overhead

The CIFS requests from a client are encapsulated as payload/attachments in SOAP message. The SOAP message is using XML syntax and results in bigger size compare to other packet format using predefined binary data format.

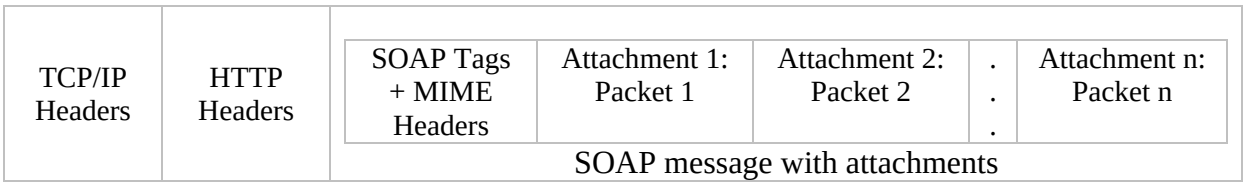

Figure 16.1 SOAP with Attachments packets

The packet put on wire includes TCP/IP and low-level protocol headers, HTTP headers. The SOAP message itself contains many MIME headers, each related to an attachment. If the SOAP messages only contain one single CIFS packets, then there will be a lot of overhead compared to sending multiple packets in one single SOAP message in addition to the processing time for the SOAP engine to process the message.

16.2 Optimization

The purpose of optimization is to improve the system to be closest as possible as CIFS traffic in the local area network. It is a difficult and impossible task to tweak the system to obtain such performance, but many enhancement techniques can be deployed to help the system to work as efficient as possible.

16.2.1 Multithreading

Multithreading enables many tasks to run simultaneously at the same time. The use of multithreading can reduce the processing time, e.g. when a SOAP message is being constructed or parsed, the system can keep reading the data from the connection. When a new message arrives, a new thread is created to perform the task or it will use existing thread (that has been created) if the requests are coming from the same client and those packets will go to the same CIFS server. This is similar as in a web server, which processes requests from different clients concurrently. For Mobile Home Access, we need another threading model as the SOAP message may contain many request packets to many different CIFS servers on the local area network. Without multithreading, a CIFS request must wait till the system has finished processing others packets.

16.2.2 Master Browser

In the CIFS protocol, network browsing may result in many broadcast packets. To eliminate such packets as well as to reduce the response time, the computer which runs Mobile Home Access server application can act as Master Browser. When remote CIFS clients request a browse list, the local CIFS server will quickly respond with the browse list instead of broadcasting it to the entire network.

16.2.3 Compress Payload

Compression algorithms can be used to compress packets to reduce the size of the SOAP message. The server then decompresses the packets and forwards them to CIFS servers. The disadvantage is that it results in higher processing time for compression and decompression of the packets. Compression can be implemented in two different places; in Mobile Home Access (as an extension) or by the application server (i.e. web server) itself.

Chapter 17 Validation

This chapter describes how well the implementation of the system has accomplished based on the requirement specification in chapter 8. This check list can be used as a source for further development and improvements to the system.

Symbol	Explanation
☺	Good
😐	Satisfactory
☹	Not well
---	Not implemented yet
?	Unknown

Table 17.1 Legend

Requirements	Status	Comments
FR1 – Authentication	---	No authentication mechanism is implemented as the authentication process can reuse from Reduced Mapping Mode.
FR2 – Authorization	---	Not implemented
FR3 – Transparency	☺	As the nature of tunneling (Tunneling Mode), users and network's resources are highly transparent.
FR4 – Administration	---	Not implemented
FR5 – Browse Service	😐	This is handled by the Tunneling Mode, but unfortunately tests could not be performed.
FR6 – Download/Upload	☺	The functionalities is fully supported.
FR7 – Mapping Resource	😐	The functionality could not be performed as long as the virtual network interface is not implemented
FR8 – Resource Manipulation	😐	Handled by the Tunneling Mode, which provides full functionality as a CIFS client on the local area network
NFR1 – Simplicity and ease	☺	The application is easy to use with intuitive GUI and the installation process is simplified using Java Archive file (JAR) and installation scripts
NFR2 – Performance	---	Unable to perform performance test because of virtual network interface dependency
NFR3 – Security	---	Out of scope of this thesis
NFR4 – Availability	?	Unable to perform the measurement as it is dependent of the number of and the time the service is running and/or fails.

NFR5 – Extensibility	☺	With Web Services and XML technologies, Mobile Home Access can be extended to support new services easily.
NFR6 – Platform independency	☺	The Web Service Architecture and Java platform provides highly platform- and network architecture independent
NFR7 - Documentation	☹	Many user guides and manuals are missing, only Java API and a limited set of guides are written

Table 17.2 Check list

Chapter 18 Future Works

The implementation of a system in this thesis is a prototype, and needs many enhancements to hit the production stage of the system. The system must go through a long process of testing to verify that the requirements are met as well as quality assurance. Many factors have prevented the author to perform such tasks, for example the lack of time and resources available made it nearly impossible to construct a complete system. This chapter will describe and propose enhancements to the system, in addition to incomplete works to be done.

Simplifying the port-forwarding process

Even the configuration of the port forwarding entry may prove too complicated for ordinary users, and solutions for simplifying the firewall/NAT traversal should be investigated and integrated in the future. This could be done by using protocols similar to STUN [62] and possibly also TCP hole punching techniques [63].

Documentation

The simple Javadoc documents generated are not sufficient enough as a complete documentation source for users and developers. Manuals and user guides must also be written in plain English for all tasks such as installation, maintenance, upgrading, etc.

Virtual network interface

Unfortunately, the implementation of the virtual network interface using TUN/TAP device did not succeed. This is due to the complexities related to the technology and because of the lack of time. The virtual network interface is crucial for the operation of the system, and the incomplete implementation has prevented the author from testing the functionality of the system throughoutly as well as implementing other functionality as packet inception from the source computer to the destination.

Switching between operation modes

The Mobile Home Access client's application installed on a computer should be able to switch between the operation modes if needed. For example when a user does not have sufficient permissions to run the virtual network interface's software application on the remote computers; the user can still access files and documents through the Reduced Mapping Mode.

Extend the set of functionality in Reduced Mapping Mode

The implementation of the Reduced Mapping Mode is, in the author's opinion, a proof-of-concept solution. Many of functionalities described in the use case diagram in figure 9.1 are not implemented, except for downloading and uploading files to the remote CIFS server. The author recommends a mechanism similar as CVS to be implemented to allow users to modify the files and documents (because many devices can now edit common file format and/or because rich clients cannot work in Tunneling Mode).

Security enhancements

One of the most important features in nowadays computer system is the degree of security. Mobile Home Access exposes the CIFS services, which are intended in the local are network, to the World Wide Web. Theoretically, everyone can access those services and this implies

that the system must implements a strong encryption/decryption mechanism to prevent unauthorized access to private resources/services. Data integrity and authenticity can be achieved using various techniques.

Recommendation of security mechanisms:

- Encryption and decryption of data using public-key cryptosystem
- Digital certificates or security hash functions to assure data integrity
- Secure the HTTP traffic (the transport protocol for Mobile Home Access web services) using Secure Socket Layer (SSL) or TLS (Transport Layer Security)

Performance enhancements

This thesis has used threads as a performance enhancement for sending and receiving CIFS packets in both server- and client-side. One of the most important performance metrics is the response time of CIFS using Mobile Home Access in Tunneling Mode. This must be compared to the response time in local area network's CIFS and in the VPN solution.

In the client application, the get web service operation is called periodically. If the clients wait for too long before sending such request, the CIFS clients on remote computer may think request-packets are lost and then retransmit those packets. This results in higher response time and a lot of retransmitted packets. On the other side, when get request is sent too often the server may not have response packets ready for retrieval. This results in higher load on Mobile Home Access server. The study of how response time can be discovered and calculated should be investigated in the future to dynamically set the periodic time value between get requests and thus, reduce the higher traffic load and the the number of retransmitted packets.

Tests enhancements

The testsuite for Mobile Home Access should also contains tests for graphical user interfaces as it is of great importance for end-users that the GUI components behave properly.

Chapter 19 Conclusion

Mobile Home Access provides a real experience of mobility in terms of accessing users' resources and services in the home networks. From a diversity of devices and terminals, home users can simply connect to computers in the home network and access them as local resources.

Short Term

With Mobile Home Access web services, we have solved many of the common problems of accessing the home network resources. For example home users do not need to setup a Virtual Private Network (which is the de-facto standard being used in large companies and organizations for connecting remote networks/sites). Mobile Home Access provides similar functionality that is not difficult to setup. In addition, accessing resources from remote devices is easily accomplished using Mobile Home Access.

Long Term

In the long run, the Mobile Home Access could be provided by the Internet Service Provider as a service that comes with an ADSL subscription. The software could be distributed as a CD with installation guidelines for home users. In addition, the Internet Service Providers can build a dynamic DNS server that resolve home users' IP addresses (as described in chapter 14).

Due to the open interfaces exposed through WSDL, it will be possible for 3rd party developers to easily implement Mobile Home Access clients for any type of connected device, e.g. for connected mp3-players, cameras etc. As opposed to building a VPN client based on e.g. IPSec, the implementation of a MHA client is quite straightforward.

References

[1] *Statistics Norway* (Statistisk sentralbyrå) http://www.ssb.no/emner/10/03/ikt/

[2] *Introduction to Mobility*, Do van Thanh, Lecture notes at NTNU 2001 http://www.item.ntnu.no/fag/tm8100/Pensumstoff2004/MOBILIThan.PPT

[3] *RFC 1918 Address Allocation for Private Internets*, Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., Lear, E., February 1996

[4] *The Mobile Home Access*, Nguyen van Tan, Research Project at NTNU on 2. semester 2005

[5] *Gnutella official homepage*, http://www.gnutella.com/

[6] *Rational Unified Process*, http://www-306.ibm.com/software/awdtools/rup/

[7] *ADPO Project Development Methodology*, Do van Thanh

[8] *Using Rational Unified Process for Small Projects*, Gary Pollice, ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/tp183.pdf

[9] *Java official homepage*, http://java.sun.com/

[10] *The Java Language Environment White Paper*, J., Gosling, H., McGilton, Sun Microsystems, May 1996

[11] *Java Native Interface*, Sun Microsystems, Inc, http://java.sun.com/j2se/1.3/docs/guide/jni/

[12] *Trail: Java Native Interface*, Stern, B., Sun Microsystems, http://java.sun.com/docs/books/tutorial/native1.1/

[13] *JDesktop Integration Components*, https://jdic.dev.java.net/

[14] *Java CIFS Client Libraryi*, http://jcifs.samba.org/

[15] *Java APIs for Web Services, JSR172*, http://jcp.org/en/jsr/detail?id=172

[16] *Java APIs for XML Processing*, Sun Microsystems, Inc, http://java.sun.com/webservices/jaxp/

[17] *Java APIs for XML-based RPC*, Sun Microsystems, Inc, http://java.sun.com/webservices/jaxrpc/

[18] *JSR101: Java APIs for XML-based RPC*, http://jcp.org/en/jsr/detail?id=101

[19] *SOAP with Attachments API for Java*, Sun Microsystems, Inc, http://java.sun.com/webservices/saaj/index.jsp

[20] *JSR67: Java APIs for XML Messaging 1.0*, http://jcp.org/en/jsr/detail?id=67

[21] *JSR110: Java APIs for WSDL*, http://www.jcp.org/en/jsr/detail?id=110

[22] *Understanding Windows Firewall*, Microsoft Corporation, http://www.microsoft.com/windowsxp/using/security/internet/sp2_wfintro.mspx

[23] *PORT NUMBERS*, IANA, September 2005 http://www.iana.org/assignments/port-numbers

[24] *RFC 1631 – The IP Network Address Translator*, K. Egevang, P. Francis, Cray Communications, NTT, May 1994, http://www.faqs.org/rfcs/rfc1631.html

[25] *RFC 2663 – IP Network Address Translator Terminology and Considerations*, P. Srisuresh, M. Holdrege, Lucent Technologies, August 1999, http://www.faqs.org/rfcs/rfc2663.html

[26] *CIFS: A Common Internet File System*, Paul Leach, Dan Perry, http://www.microsoft.com/mind/1196/cifs.asp

[27] *Implementing CIFS*, Christopher R. Hertel, http://www.ubiqx.org/cifs/

[28] *RFC 3530 - Network File System version 4 Protocol*, Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, April 2003, http://www.faqs.org/rfcs/rfc3530.html

[29] *RFC 1001 – Protocol standard for a NetBIOS service on a TCP/UPD transport: Concepts and methods*, March 1987, http://www.faqs.org/rfcs/rfc1001.html

[30] *RFC 1001 – Protocol standard for a NetBIOS service on a TCP/UPD transport: Detailed Specifications*, March 1987, http://www.faqs.org/rfcs/rfc1002.html

[31] *RFC 2141 - URN Syntax*, R. Moats, AT&T, May 1997, http://www.faqs.org/rfcs/rfc2141.html

[32] *SMB File Sharing URI Scheme*, Christopher R. Hertel, http://www.ietf.org/internet-drafts/draft-crhertel-smb-url-10.txt

[33] *CIFS/E Browser Protocol*, Paul Leach, Dilip Naik, Microsoft Corporation, January 1997, ftp://ftp.microsoft.com/developr/drg/cifs/cifsbrow.doc

[34] *Official SAMBA site*, http://www.samba.org/

[35] *Extensible Markup Language (XML) 1.0*, W3C Recommendation, February 2004, http://www.w3.org/TR/REC-xml/

[36] *Web Service Architecture*, W3C, February 2004, http://www.w3.org/TR/ws-arch/

[37] *Web Services in a Service-Oriented Architecture*, https://bpcatalog.dev.java.net/nonav/soa/index.html

[38] *Simple Object Access Protocol*, W3C Recommendation, http://www.w3.org/TR/soap/

[39] *XML Schema*, W3C, http://www.w3.org/XML/Schema

[40] *SOAP Part 1: Messaging Framework*, W3C Recommendation, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

[41] *SOAP Messages with Attachments*, W3C, December 2000, http://www.w3.org/TR/SOAP-attachments

[42] *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, N. Freed, Innosoft, N. Borenstein, First Virtual, November 1996, http://www.ietf.org/rfc/rfc2045.txt

[43] *RFC 2387 – The MIME Multipart/Related Content-type*, E. Levinson, August 1998, http://www.ietf.org/rfc/rfc2387.txt

[44] *Direct Internet Message Encapsulation (DIME) Specifications*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/dimeindex.asp

[45] *Sending Files, Attachments, and SOAP Message via DIME*, Jeannine Hall Gailey, http://msdn.microsoft.com/msdnmag/issues/02/12/DIME/

[46] *Web Services Description Language (WSDL)*, Christensen, E., Franscisco, C., Meredith, G., Weerawarana, S., W3 Consortium, March 2001, http://www.w3.org/TR/wsdl

[47] *Which style of WSDL should I use?*, Russel Butek, IBM, May 2005, http://www-128.ibm.com/developerworks/webservices/library/ws-whichwsdl/

[48] *Universal Description, Discovery and Integration*, OASIS, http://www.uddi.org/specification.html

[49] *Universal Plug and Play Forum*, http://www.upnp.org/

[50] *Enabling Mobile Access to the User Home Network*, Jørstad I., van Do, Thanh, van Do, Thuan, 12th International Conference on Telecommunications (ICT2005), Cape Town, South Africa, 03-06 May 2005, ISBN: 0-9584901-3-9

[51] *Concurrent Versions System*, http://www.nongnu.org/cvs/

[52] *Apache Ant*, http://ant.apache.org/

[53] *Eclipse Project*, http://www.eclipse.org/

[54] *Eclipse C/C++ Development Tooling – CDT*, http://www.eclipse.org/cdt/

[55] *Apache Tomcat*, http://tomcat.apache.org/

[56] *JSR53: Java Servlet 2.3 and JavaServer Pages 1.2 Specifications*, http://jcp.org/aboutJava/communityprocess/final/jsr053/

[57] *Apache Axis*, http://ws.apache.org/axis/

[58] *Ethereal Network Analyzer*, http://www.ethereal.com/

[59] *Gentoo Linux*, http://www.gentoo.org/

[60] *TUN/TAP Device*, http://vtun.sourceforge.net/tun/

[61] *JUnit*, http://www.junit.org/

[62] *RFC 2489 – STUN – Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)*, J. Rosenber, J. Weinberger, C. Huitema, R. Mahy, dynamixsoft, Microsoft, Cisco, March 2003, http://www.faqs.org/rfcs/rfc3489.html

[63] *Peer-to-Peer Coummnication Across Network Address Translators*, Bryan Ford, Pyda Srisuresh, Dan Kegel, http://www.brynosaurus.com/pub/net/p2pnat/

Appendix A Setup web services environment with Tomcat and Axis

A.1 Tomcat web server

Before you start, your system must have Java Runtime Environment (JRE) installed. Otherwise, you can download a copy at http://java.sun.com/j2se (current release: JRE 5.0 Update 5). After installing JRE, be sure to set or check that the environment variable named JAVA_HOME is set to the pathname of the directory into which you installed the JRE.

Download and install Tomcat

Even though there exist, for Windows users, windows executable distribution of Tomcat. The author chooses to get a zip-package and installs it manually. The reason is to be able to generate easily an automatic installation script for the MHA later if we understand the installation procedure.

Download and installation of Tomcat includes the following steps:

1. Download binary(zip) package from http://jakarta.apache.org/tomcat/download-55.cgi
2. Unzip the files into a convenient location for example C:\jakarta-tomcat-5, the location is from now on referred as $CATALINA_HOME
3. Open the command prompt and run the startup script *startup.bat* (Windows) or *startup.sh* (Unix) at $CATALINA_HOME\bin (note: Unix system uses forward slash /)
4. If the startup script run successfully, open your web browser and then type http://localhost:8080/
5. If you can see a window similar as the figure below, then the installation is completed

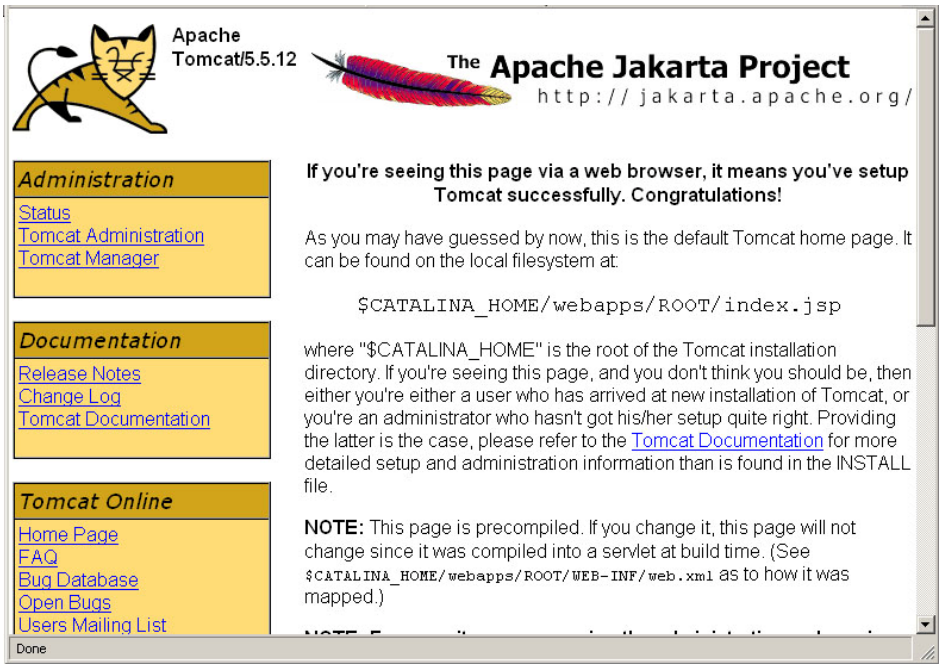

Figure A.1 Tomcat welcome page

Change port number
Tomcat runs on port 8080 by default. If you have other application running on the same port, Tomcat will fail during startup. In that case, you must change to a port number that is not in use for Tomcat to run properly. This is done by editing the configuration file, server.xml, in $CATALINA\conf directory.

Run Tomcat as Windows service/Unix daemon
To install Tomcat as service has many benefits. First, you can modify it to start the service automatically on boot/reboot. Second, you can use the monitor and system tray service. Installing Tomcat as Windows service use the script provided, i.e. *service.bat* in the $CATALINA_HOME\bin directory.

1. Open command prompt and go to $CATALINA_HOME\bin directory
2. Type "service.bat install" and press Enter
3. You can now open Service Manager (Start->Run : services.msc) and change the startup type (manual, automatic, disable)

Unix daemon can be installed using the jsvc tool from the commons-daemon project. The source is included in $CATALINA_HOME/bin folder and need to be compiled.

Compile source:
```
autoconf
./configure
make
```

Run tomcat as a service daemon:
```
jsvc -Djava.endorsed.dirs=./common/endorsed -cp ./bin/bootstrap.jar \
        -outfile ./logs/catalina.out -errfile ./logs/catalina.err \
        org.apache.catalina.startup.Bootstrap
```

Tomcat Administration and Manager
To be able to access Tomcat Adminstration and Manager, edit the configuration file *tomcat-users.xml* in the $CATALINA_HOME\conf directory.

1. Add admin and manager role.
   ```
   <role rolename="admin"/>
   <role rolename="manager"/>
   ```
2. Add new user and assign role(s), for example
   ```
   <user username="admin" password="secret" roles="admin,manager">
   ```

A.2 Axis

1. Download zip package from Axis from http://ws.apache.org/axis/
2. Unpack the file download, and copy the whole *axis* directory in the webapps directory to $CATLINA_HOME\webapps
3. Restart Tomcat web server
4. Test Axis installation by http://localhost/axis, you should see a welcome page similar to the figure below.

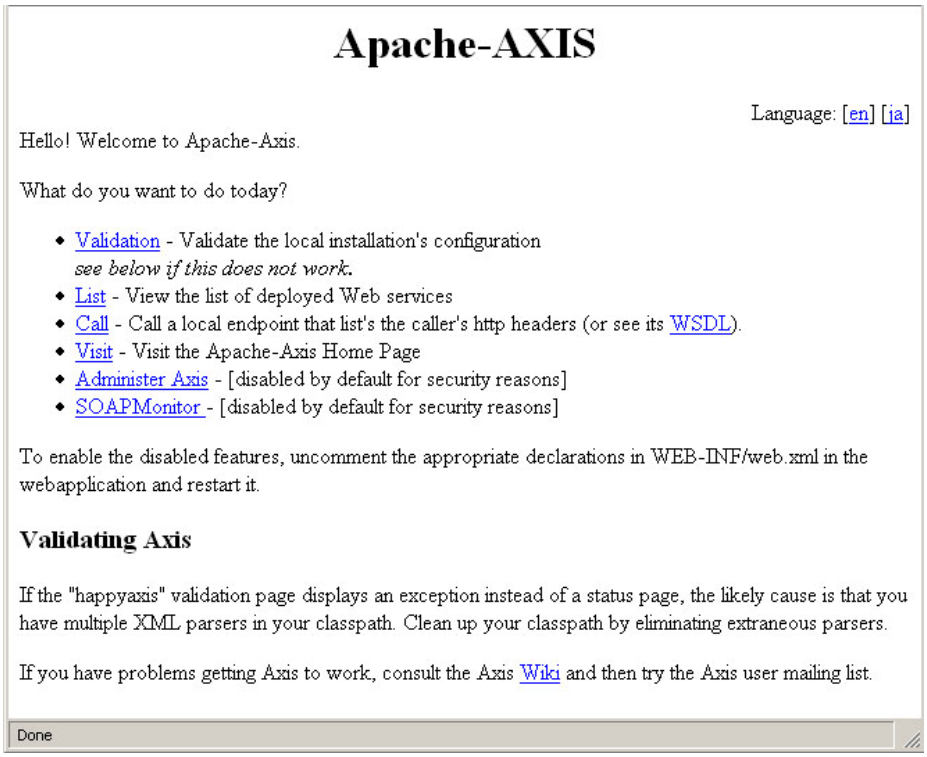

Figure A.2 Axis welcome page

Validation and add optional libraries

To validate the installation of Axis, click the *Validation* (i.e. happyaxis.jsp) link in the figure above. At this point, the author got one missing library, *activation.jar* and two warnings for optional libraries, *mail.jar* and *xmlsec.jar*. Fortunately, Axis provides links where to download these libraries.

1. Follow the link and download the zip packages
2. Unpack and add the **.jar* files into $CATALINA_HOME\webapps\axis\WEB-INF\lib directory

Due to the Axis team recommendation, Xerces is chosen as the XML parser.

1. Download Xerces library from http://xml.apache.org/dist/xerces-j/
2. Add *xml-apis.jar* and *xercesImpl.jar* to $CATALINA_HOME\webapps\axis\WEB\lib directory

Restart Tomcat web server and refresh the validation page. Now Axis is happy☺

Enabling SOAP Monitor

1. Set environment variables for Axis. This is the content of *axisCP.bat* script:
```
set AXIS_HOME=%CATALINA_HOME%\webapps\axis
set AXIS_LIB=%AXIS_HOME%\WEB-INF\lib
set AXISCLASSPATH=%AXIS_LIB%\axis.jar;%AXIS_LIB%\commons-discovery-
0.2.jar;%AXIS_LIB%\commons-logging-
1.0.4.jar;%AXIS_LIB%\jaxrpc.jar;%AXIS_LIB%\saaj.jar;%AXIS_LIB%\log4j-
1.2.8.jar;%AXIS_LIB%\xml-apis.jar;%AXIS_LIB%\xercesImpl.jar
```
2. Compile SOAP Monitor in $CATALINA_HOME\webapps\axis
```
javac –cp %AXIS_LIB\axis.jar SOAPMonitorApplet.java
```
3. Deploy the service using the deployment descriptor at Axis homepage
```
java –cp “%AXISCLASSPATH%” org.apache.axis.client.AdminClient –
lhttp://locahost/axis/services/AdminService deploy-monitor.wsdd
```

4. Add service to be monitored by adding these lines in the deployment descriptor (between the *service* tag) of the service.
```
<requestFlow>
   <handler type="soapmonitor"/>
</requestFlow>
<responseFlow>
   <handler type="soapmonitor"/>
</responseFlow>
```
5. To enable SOAP Monitor for all web services edit the *server-config.wsdd* file in $AXIS_HOME\WEB-INF directory, and between <*globalConfiguration*> tag add the handlers for request and response flow similar to the previous step.
6. Access the SOAP Monitor at http://localhost:8080/axis/SOAPMonitor

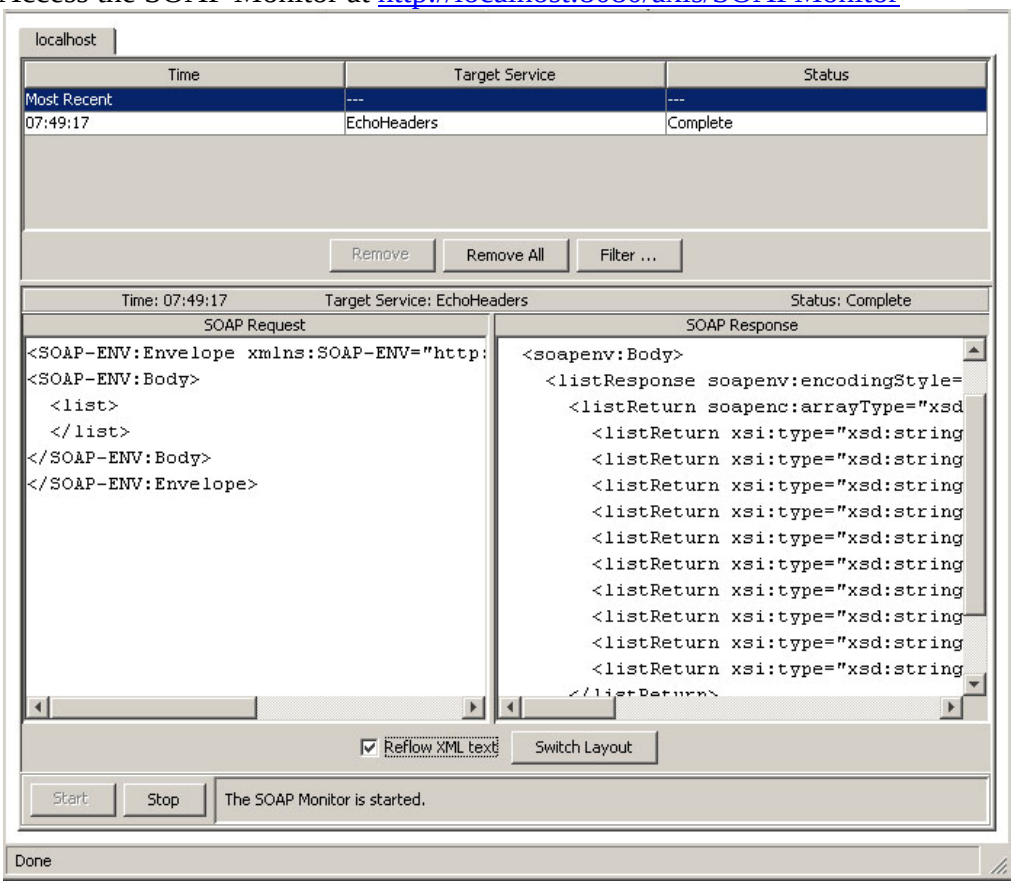

Figure A.3 SOAP Monitor

Enable TCP Monitor

1. To setup TCP Monitor to listen at port 8011 and redirect the request to Tomcat server (at port 8080), then in command prompt type:
```
java –cp "%AXISCLASSPATH%" org.apache.axis.utils.tcpmon 8011
localhost 8080
```
2. From this point forward, to monitor the requests/responses, point to the listener port of TCP Monitor, i.e. 8011.
http://localhost:8011/axis/

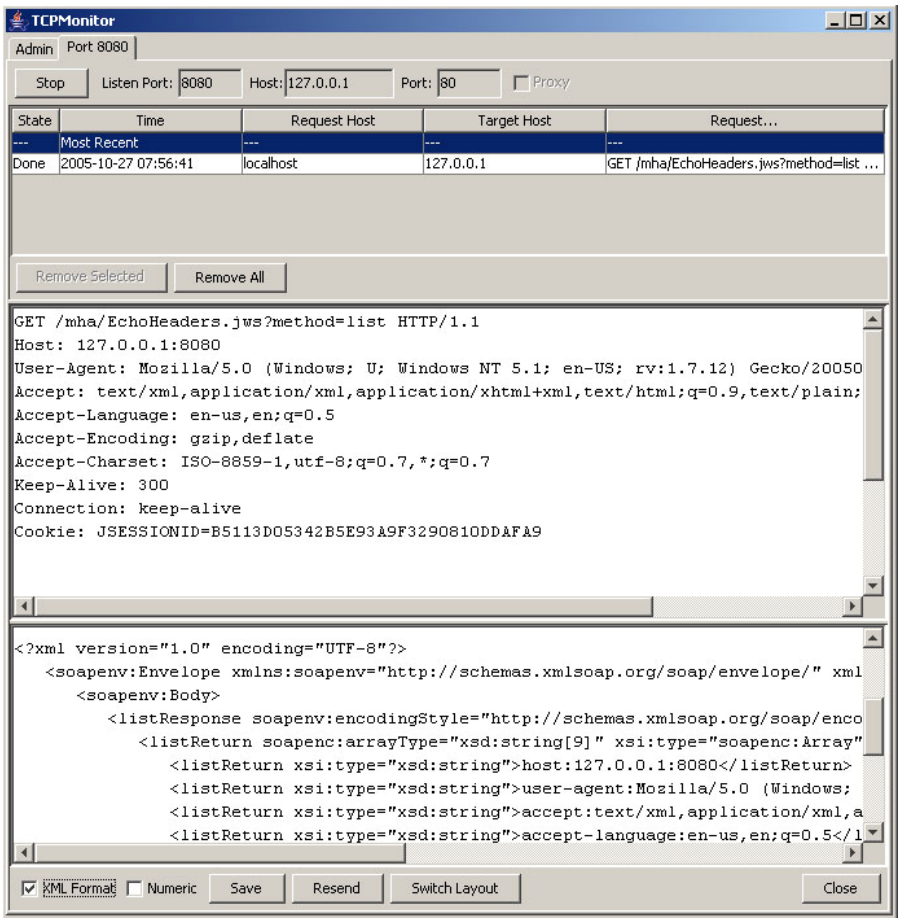

Figure A.4 TCP Monitor

Appendix B Setup the home network

B.1 Setup Windows Network (for Windows XP)

Use the *Network Setup Wizard* to set the settings for network access, such as computer name and workgroup name. We can also use the *Network Identification Wizard* to set the computer name.

File and Printer Sharing for Microsoft Network must be enabled. View how to turn on Simple File Sharing: http://support.microsoft.com/servicedesks/ShowMeHow/304040.asx

Tips

- Ensure each workgroup and domain name is no longer than 15 characters, and do not contain spaces or special characters such as / \ * , . " @.
- Avoid using lower-case letters in workgroup or domain names.
- Ensure that all computers on the LAN use a unique computer name.
- To make network browsing easier, use the same workgroup/domain in all computers on the LAN.

B.2 Utilities

There are a few utilities that may be useful when troubleshooting or to view the network state. Here are some examples:

nbtstat

-n	- List your NetBIOS names
-s	- List your current NetBIOS sessions

netstat

-a	- Displays all listening ports and active connections
-n	- Displays addresses and port numbers in numerical form

net

view	- Lists available computers with NetBIOS support
view *\\computername*	- List visible shares for a specific computer.

B.4 Troubleshooting Windows Network

Windows network (workgroup) is great as long as it works. Unfortunately, there are many pitfalls users may run into that add frustrations and heavy load on Windows support as well as forums on the Internet. On Windows XP, the easiest way to configure Windows is using Network Setup Wizard.

***Workgroup Name* is not accessible**

This problem is described in Knowledge Base 318030 (http://support.microsoft.com/kb/318030/) .

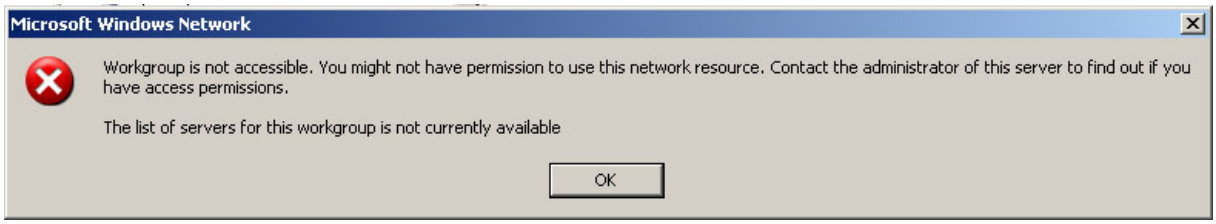

Figure B.1 Not Accessible Error Message

Solution, for each computer in workgroup:

- Enable NetBIOS over TCP/IP
- Turn on Computer Browser Service

LAN Subnets

There are situations where users may have multiple routers and therefore it is also possible that there are subnets. For example, users may have both wired and wireless router. In that case, network browsing in *My Network Places* does not list all computers on the LAN.

The solution is to apply the correct IP address instead of server (or NetBIOS) name of a specific computer when browsing or mapping a network drive.