



**DUBLIN CITY UNIVERSITY
SCHOOL OF ELECTRONIC ENGINEERING**

**Unified access to 3G
and WLAN-based Local services**

Michael Goulding
August 2005

MASTER OF ENGINEERING

IN

Telecommunications

Supervised by Dr. S. Murphy

Acknowledgements

I would like to thank my supervisor Dr. Sean Murphy for his guidance, enthusiasm and commitment to this project. The regular meetings helped greatly to keep the goals and issues of the project clear.

Declaration

I hereby declare that, except where otherwise indicated, this document is entirely my own work and has not been submitted in whole or in part to any other university.

Signed:

Date:

Abstract

This project investigates the concept of unified access to WLAN and 3G local services in a location based system. A system architecture for a mobile client and a web server, was designed and developed to examine this. As this system was required to be location aware an overview of location based services and location estimation techniques are given. The difficulties encountered when developing for a mobile phone running the Symbian C++ environment along with an overview of the system architecture are also reported. Communication to retrieve the service objects was done using a SOAP implementation which was straight forward on the server side but not so on the client. The end result proved to be successful the client retrieved data from the application servers using both the WLAN and 3G network.

Table Of Contents

ACKNOWLEDGEMENTS	2
DECLARATION	2
ABSTRACT	3
TABLE OF CONTENTS	4
TABLE OF FIGURES	6
CHAPTER 1 - INTRODUCTION	7
1.1 BACKGROUND	7
1.2 PROJECT OBJECTIVE	8
CHAPTER 2 – LOCATION BASED SERVICES	9
2.1 WHAT ARE LOCATION BASED SERVICES	9
2.2 LOCATION DETERMINATION	11
2.2.1 <i>Location determination in GSM and UMTS Networks</i>	11
2.2.3 <i>Location determination in WLAN Networks</i>	12
2.3 LOCATION BASED APPLICATIONS	13
2.4 PARLAY APIS	13
2.5 SUMMARY	14
CHAPTER 3 – SYMBIAN APPLICATION DEVELOPMENT	15
3.1 MOBILE DEVELOPMENT	15
3.2 SYMBIAN OS FUNDAMENTALS	16
3.3 OS STRUCTURE	16
3.4 APPLICATION FRAMEWORK	17
3.5 DIFFICULTIES WITH SYMBIAN C++ DEVELOPMENT.....	18
CHAPTER 4 – SYSTEM ARCHITECTURE	20
4.1 GOALS AND ISSUES.....	20
4.2 OVERVIEW OF SYSTEM	21
4.3 TECHNOLOGIES USED	22
4.3.1 <i>Client Design Considerations</i>	22

4.3.2	<i>Server Design Considerations</i>	23
4.4	SOAP.....	23
4.5	SERVER DISCOVERY.....	24
CHAPTER 5 – IMPLEMENTATION AND DESIGN		25
5.1	HARDWARE.....	25
5.3	CLIENT ARCHITECTURE.....	26
5.3.1	<i>Client Connection handling</i>	26
5.3.2	<i>SOAP on client</i>	27
5.4	SERVER ARCHITECTURE.....	28
	FOR THIS PROJECT THE DATA RETURNED WAS FIXED I.E. NO DATABASE QUERYING WAS PERFORMED. BOTH THE LOCAL AND THE REMOTE APPLICATION SERVERS RAN THE SAME CODE.....	28
5.5	SOAP IMPLEMENTATION DETAILS.....	28
CHAPTER 6 – RESULTS ANALYSIS		31
6.1	RESULTS AND DESIGN ANALYSIS.....	31
CHAPTER 7 – CONCLUSIONS AND FUTURE WORK		33
7.1	FUTURE WORK.....	33
7.2	CONCLUSIONS.....	34
REFERENCES		35
APPENDIX 1		36
GLOSSARY		37

Table of Figures

Figure 3.1: Symbian series 80 OS.....	16
Figure 4.1: Overview of system architecture.....	21
Figure 5.1: Localserv Application Structure.....	26
Figure 5.2: Server Architecture.....	28

Chapter 1 - Introduction

With the growth in wireless local area network hotspots and the introduction of next generation public wireless networks like 3G, service providers are striving to introduce web services that take advantage of this new increased wireless network capacity. Coupled with this is the release of new high capability mobile handsets called smartphones which offer the mobile developer increased computing power and resources. Location Based Services (LBS) is just one of the new breed of services that are being introduced to utilize these new technologies. LBS provide location aware services for the mobile client and are a major area of interest in the wireless domain. In this project the provision of local services over both a wireless local area network (WLAN) and over the 3G network is investigated.

1.1 Background

Currently the two main technologies for providing high speed wireless connectivity to a mobile station are 3G and WiFi. 3G or UMTS (Universal Mobile Telephone Service) refers to the collection of third generation mobile technologies that are designed to allow mobile operators offer integrated data and voice services over mobile networks. WiFi refers to the 802.11 series of wireless Ethernet standards that are designed to support WLAN. There has been a lot of debate as to which of these two technologies would become the dominant system for wireless broadband. The characteristics of the service provided by these two technologies are contrasting. 3G networks provide always on connectivity with relatively low data rates but high mobility to users. WiFi networks provide high data rates but can only cover users with low mobility in smaller areas. These complementary characteristics make it attractive to combine the two technologies.

From a location based services perspective a user could take advantage of downloading at high speeds while in the area of a WLAN and take advantage of 3G and it's widespread connectivity when travelling around. Combining the two also offers the user local services that can be returned from both networks. Local services that are hosted by, for example a restaurant, could be made available over the WLAN, while wider area services like 'route finders' could be made available over the 3G network. Both local services would

complement each other. The difficulties this presents is that a mobile handset would have to connect to both networks and retrieve the correct services based on it's location.

Mobile handsets that have both WLAN and 3G network connectivity are available, with several more planned releases in the near future. What are needed are applications that can make use of the dual network capabilities of these handsets and LBS is one such system that can provide these.

1.2 Project Objective

The objective of this project is to design and implement a software architecture that enables the access of local services provisioned over the 3G and WLAN networks. This software architecture includes both the mobile client, application server and the communication between the two. Developing in the mobile handset environment and creating a successful communications system provide the main challenges.

Chapter 2 – Location Based Services

This chapter gives a description of location based services and the elements needed to provide a LBS system. LBS are amongst the fastest growing services in mobile communications and offer a large variety of new and exciting applications for the mobile user. In order for LBS to be successful the location estimation must be accurate. The techniques that can be used to determine location in the public mobile network and a wireless LAN network are discussed. Examples of LBS applications that are currently being deployed are mentioned along with the emerging Parlay API.

2.1 What are location based services

Location based services use a requestor's location in order to provide the best response to a request. With the mobile phone boom, introduction of the new generation of sophisticated smart phones and position tracking technologies, location based services have become the most rapidly expanding field in mobile communications [2]. These location based response systems offer the network provider the ability to enhance the services and applications that can be offered to the mobile user. Video clips, ring tones, images and information content that can be targeted to a customer's location. The market projections for this business indicate revenues of over 10 billion USD by 2005 [3]. Today, the mobile data services offered by network providers are greatly underused as customers are either unaware or the usability is poor of these services.

Location based services have an objective "To assist with the exact information, at the right place in real time with personalised setup and location sensitivity" [4]. Typically a LBS system consists of a server which can analyse a users request and generate a reply based on the user's location. Consider the following examples:

- User is in a coffee shop around Grafton Street and is looking for someplace to eat. Information like the menus of nearby restaurants, special offers and if there are any tables free would be of interest. By utilizing a LBS application on his/her mobile handset and selecting 'food' the user could be presented with a list of local eateries.

From here the user could make further selections to retrieve the information he/she desired.

- User is in same coffee shop but is interested in finding out the nearest theatre and the plays that are currently running. By again using a LBS application and selecting the appropriate links the user would be presented with the address of e.g. the Gaiety and it's current running list.

Taking the example above and looking more closely at the client-server interaction, some of steps taken by such a system might be:

- The application is started and registered itself with local and remote LBS server.
- The user's location is determined.
- The request to find the local restaurants is sent to the local server which returns it's locally stored data.
- Using the user's location the address and details of the nearest theatre is returned to the user from the remote LBS server.

For such an example it might be convenient to retrieve the local information from the restaurant's own menu server or a server hosted within the local area. Ease of provisioning and editing of the content and more current content would be the advantages. For such a system, and for the provisioning of other local services, the use of a local WLAN hotspot would be advantageous. When the local services desired require information from a remote server, using the public mobile phone to retrieve the data works better. In our example the list of local theatres is a large area, city wide query that a local WLAN hotspot might not be able to fulfil accurately.

The combination of querying the local WLAN hotspot and the remote public mobile network servers for local services will provide the most desired results. However, for this system to work best the location of the user needs to be known. In the WLAN situation it's implicitly known as the user must be within range of the WLAN in order to communicate. In a public mobile phone network like the 3G UMTS network the user's position has to be calculated.

2.2 Location determination

In order to provide the most relevant location based services the accuracy of the positioning calculation must be high. For the wireless hotspot LBS the users position range is known and no greater accuracy is usually needed. On the UMTS side the accuracy of the position typically depends on the cell size. Inside a city this the accuracy might vary from 100 to 500 meters, however in a countryside cell this accuracy could be up to 30 Kilometres.

This accuracy depends on the infrastructure on the network side, technology used and the time provided to determine the position [5]. For great accuracy investment in new UMTS network elements and possibly investment in the handset is needed. The technologies used to determine position can be grouped into the following groups:

- Satellite based positioning – an example of this technology is GPS (Global Positioning System) which can provide accuracies of up to 5 meters. The main disadvantage of these systems is that they require handset modification and they do not work well indoors.
- Network positioning methods – These methods use existing network infrastructures with additional positioning elements with no modification to the handsets required. The accuracy of such systems depends on the size of the mobile network cell.
- Local positioning methods – such as indoor sensors and location systems for wireless LANs provide high accuracy but limited range.

Some systems consist of two or more of the above technologies working together to provide a more accurate location determination. The degree of accuracy needed is depended on the service being provided – mapping and route finder services need high accuracy and rapid calculation response whereas a cinema finder can be less accurate as there are relatively few per large geographical area.

2.2.1 Location determination in GSM and UMTS Networks

Location determination of mobile equipment usually applies two main principles: measurement of the signal and the computed location estimated using this measurement. Standards have specified two location determination implementations based on which

network element calculates the position of the mobile handset. In network based mode the mobile equipment takes the signal measurement and relays this to the network for position calculation. In the mobile equipment based mode, it makes the measurements and also calculates the position with some assistance data from the network.

In current networks there is support for basic positioning methods based on Cell Identifier (CI) and Timing Advance (TA) techniques, which can give an accuracy of hundreds of meters. The CI method estimates the position of the mobile handset is based on the network cell it's in. The accuracy of this reading obviously depends on the size of the cell, which can vary greatly from city to the rural size. Timing Advance can be added to the CI method to improve the calculation. This method estimates the distance to the mobile handset from the cell node from the received signal. Combining the two will give an arc range within a cell where the handset is calculated to be.

In 2.5G (GPRS) and 3G systems elements for further advanced positioning techniques have been added to give a more accurate location calculation. These advanced techniques like OTDOA-UMTS (Observed Time Difference on Arrival) and EOTD-GSM (Estimated Observed Time Difference), calculate location based on the cell's downlink signals measured at the mobile station and assistant measurement units, to provide an accurate estimation on location. The standards for these location based services are being worked on by many organisations such as OMA (The Open Mobile Alliance), MLP (Mobile Location Protocol), IETF (Internet Engineering Task Force) and OGC (Open GIS Consortium).

2.2.3 Location determination in WLAN Networks

Though not generally needed for LBS systems there are a number of methods to determine the location of a user in a WLAN. The fundamental and challenging problem for WLAN location based application is to sensing and tracking mobile users [6]. The two main types of location-sensing for indoor environments are propagation-based and location fingerprinting techniques. In the propagation-based technique the received signal strength, the angle of arrival and the time difference of arrival of the received signals are used to calculate the user's location. The fingerprinting technique uses the fact that the received signal at different locations will have different characteristics (fingerprints) like the received signal strength. The data is collected and stored for different locations and used to compare with the current fingerprint to determine the location. This method is more accurate than the

propagation-based technique but both suffer from sensitivity to changes in the environment (e.g. temperature increase) which can affect the signal characteristics.

2.3 Location based applications

Location based services can be classified into the following groupings: Emergency Services, Informational Services, Tracking and Entertainment Services. The required accuracy and the calculation time for the location estimation differ for each group.

These LBS can provide either pull or push based services. In a pull based service the user initiates query for local information e.g. “Find nearest restaurant” – whereas in a push based service the information is ‘pushed’ to the user without request e.g. “Traffic alerts”. All pull based LBS user client have the following functionality:

- Calculate or retrieve position.
- Query LBS server and include position estimation.
- Retrieve results and display to user.

Examples of location based services that are on offer from different network operators include:

- Search facilities to find taxis, hotels, cash dispensers, gas stations, etc..
- Shopping, traffic and weather information.
- Location finding services to find other users, vehicles or field agents.
- Route planners and yellow pages.

2.4 Parlay APIs

The Parlay application programming interfaces (API) offer the network application developer access to the network provider’s core network resources. The rationale behind the Parlay API is that attracts innovation from third parties that are outside the network operator’s domain to build and deploy new network applications [7]. These applications will have the ability to interact with network functionality such as location service (amongst others). The network operator would still own and manage these services but they would open them up for whatever use deemed appropriate. Using the location service estimation of a handsets location an independent third party companies could offer there own location

based services. With third party companies developing LBS the growth and variety of the services being offered will increase greatly.

2.5 Summary

Location based services offer the mobile handset user a new range of applications that can customize services for the location of the user. LBS are amongst the fastest growing services in the mobile arena and are already deployed by some network operators. In order for the LBS service to excel the estimation of the users location must be accurate. Several techniques can be used for this calculation with the more accurate OTDOA method requiring extra network elements to help improve the estimation. Although not needed for the LBS in this project, location estimation can be performed in WLAN networks as well. The Parlay APIs might offer independent companies the abilities to use these location estimations to provide further services.

Chapter 3 – Symbian Application Development

Mobile application development and the Symbian operating system present a new and challenging programming environment to the developer. The level of object oriented design and C++ employed throughout the Symbian OS result in a significant learning curve compared to other environments. In this chapter and introduction the mobile development and the Symbian OS will be given. While not going too deep into the Symbian OS the standard framework for application development will be described. The difficulties that were experienced in getting familiar with this environment will also be presented.

3.1 Mobile development

With the boom in popularity of mobile phones and the introduction of advanced handsets (smartphones), that have sophisticated operating systems and increased hardware specifications, developers are now able to develop complex applications to run on these handsets. With full functionality operating systems and relatively large storage space, these applications are becoming easier build. The community of mobile handset developers is growing and these applications are breaking into the general public market space. Currently, packages that help consumers manage their personal and professional lives are in widespread use. Soon, service providers will be offering rich media services and giving users the ability to purchase them, with the goal of increasing the use of data traffic in the network. Symbian is currently the most popular handset operating system with full support from handset manufacturers like Nokia and Sony Ericsson.

However, unlike developing for the PC, developing for the smartphone puts considerable constraints on the developer such as:

- **Hardware** - There is limited hardware resources on handsets compared to PCs – the typical smartphone will have 220Mhz processor, 50MB ROM, 30MB RAM and 64MB memory card.
- **Reliability** - The mobile phone is a very personal possession and owners will not accept the PC scenario of having to reboot their phone everyday. Added to this is

the fact that some phones will not be switched off for months and maybe even years. 100% reliability is a very important and critical goal for application developers.

- Asynchronous events – Events like phones calls, text messages and handset power off can occur at any time during handset operation and the application must be able to handle these asynchronous events.

With these constraints in mind the Symbian OS was developed.

3.2 Symbian OS Fundamentals

The Symbian OS (formerly called EPOC) is a 32-bit, little-endian, pre-emptive multitasking operating system written mainly in C++. From the beginning the Symbian OS was designed to run on limited battery powered devices used for communications purposes. It is designed to sophisticated, powerful and very reliable [8]. Some of the key design features include: good device power management, multitasking, object oriented software, use of industry standard technologies, runtime memory requirements minimised and memory management optimized for embedded software environment.

3.3 OS Structure

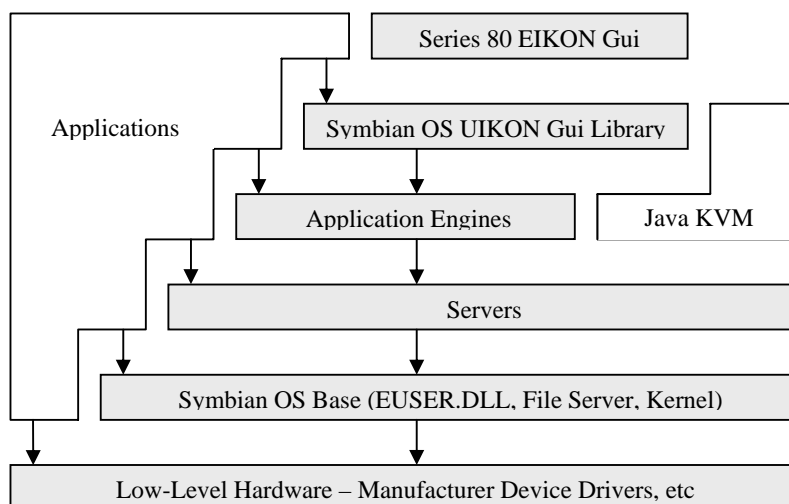


Figure 3.1: Symbian series 80 OS [8]

The Symbian OS Base layer contains the system kernel, file server, memory management and device drivers. The kernel manages system resources such as memory. The developer can access base operating system functionality through the EUser libraries. The upper layers provide the communications and computing services with the Eikon layer defines interface components and the application framework components. From Figure 3.1 it can be seen that Symbian applications can access all but the lowest layer.

Client/server architecture is key in the design of Symbian OS. The system servers provide resources for all the client applications like system processes and user applications. These servers run with a high priority and can only be accessed by the clients through well defined interfaces.

The Series 80 platform is a complete smartphone reference design developed by Nokia Mobile Software [8] which builds on the OS from Symbian adding configurable graphical user interface library and a suite of applications plus other general purpose engines. Other platforms include Series 60, Series 90 and UIQ. In this project the series 80 platform for the Nokia Communicator handset was used.

3.4 Application Framework

A framework provides the developer with a template like software system where abstract base classes and ready-made concrete classes are defined. Using this, the developer derives new implementations of the abstract classes, providing new behaviour and makes use of the default implementation otherwise. In Symbian OS there is the concept called the “application framework”. This refers to the set of classes that need to be implemented in order for an application with a decent-looking user interface (UI) to work. There is also the option of developing an application with a basic, command-line like UI, called a console application which does not require this application framework.

The four main application framework classes are:

- **Application:** This class defines the application properties and operates as a start-up object. It’s base class is `CEikApplication` and it has two main purposes: to return the applications unique identifier and to create the next object in the framework, the document object.

- Document: This object is used to store the applications persistent state and all applications must have an instance of the document class even if it's required to do nothing else other than to create the application UI object. The base class for documents is `CEikDocument`.
- Application UI: This class is derived from `CEikAppUI` and is responsible for handling application wide events such as application focus, options menu commands and opening closing/files. It can own a number of Views and delegates drawing/screen based interaction to these views.
- View: The view controls the display of data on the screen and is closely tied to the Application UI. It is also used to collect and return to the UI, user input. A view is derived from either `CEikDialog` or `CCoeControl`.

In a standard Symbian application there is also the concept of a Model/Engine. This encapsulates the application functionality such as data persistence and algorithms and also allows developers to separate the UI and functional parts of an application. Usually the Application UI constructs and owns the Model/Engine. Below is the application start-up sequence which shows when the classes are created:

- New thread is created and the Uikon (`CEikonEnv`) environment created in this thread.
- The Uikon environment loads the application DLL and calls `NewApplication()` (Application class).
- Application created and `CreateDocument()` called to create the document.
- Uikon then calls `CreateAppUiL()` to create App UI. In the App UI constructor the App View is created.
- Control environment takes over and waits for user input

3.5 Difficulties with Symbian C++ development

Symbian programming for mobile handsets is different than any other C++ object oriented environment and it requires the developer to go through a slow and steep learning curve before any kind of results can be obtained. The lack of decent API documentation, example applications and help adds to this struggle. There is also poor IDE integration with most just being glorified text editors.

Implementing the frameworks adds complexity to the code - even relatively simple functionality requires a large amount of code to be written. The standard Hello World basic application consists of 14 files with an average of between thirty to forty lines of code in each. This application only prints hello world text to the screen when a menu is selected. Developers also have to be aware of the new data types that Symbian C++ introduces to help reduce resources needed on handsets. The most confusing of these is the string descriptor classes and it's not obvious at all how to use these. There is no garbage collector in Symbian and additionally you must manually manage the lifetime of all objects

Chapter 4 – System Architecture

4.1 Goals and Issues

The goal of this software solution is to prove the concept of a mobile handset application which is able to deliver local services retrieved over the UMTS and WLAN networks. This design should include both a client and server side for both the local WLAN and the remote UMTS systems. The mobile client should be able to seamlessly query the local server using a WLAN network and the remote servers using either the UMTS (or GPRS network). Content from both servers should be presented to the user in a consistent way with only UI elements used to distinguish the origins of the data.

The client application design should strive to use as much common code as possible between the local WLAN and the remote UMTS queries. The structure of the queries and the returned data should also be the same for both. For good mobile client application design the client engine, which provides the functionality, and the UI should be separated. This allows the client to be ported to different handset types in the future. As it's a 'proof of concept' design actual local services content and UI presentation are not of high priority.

The client has to discover the local and remote content servers. Discovering the remote server is made straight forward by the 'always connected' characteristic of the UMTS network meaning that this remote server can stay fixed. But for the local server it's not so straight forward – a user will roam into various independent WLAN hotspots with different service providing servers, all of whom have to be discovered and connected to in order to get local content.

The client-server system must be designed so that querying for the required service is efficient. The user will firstly query for a list of hosts, from a host a list of service providers, from a service provider a list of services and finally from a service the service object. The service object could be anything from a menu price list to a functional object. The query response dialog between client and server must be able to handle this. The following

solution presents the technologies that were used to implement this design and informs of the issues that were discovered.

4.2 Overview of System

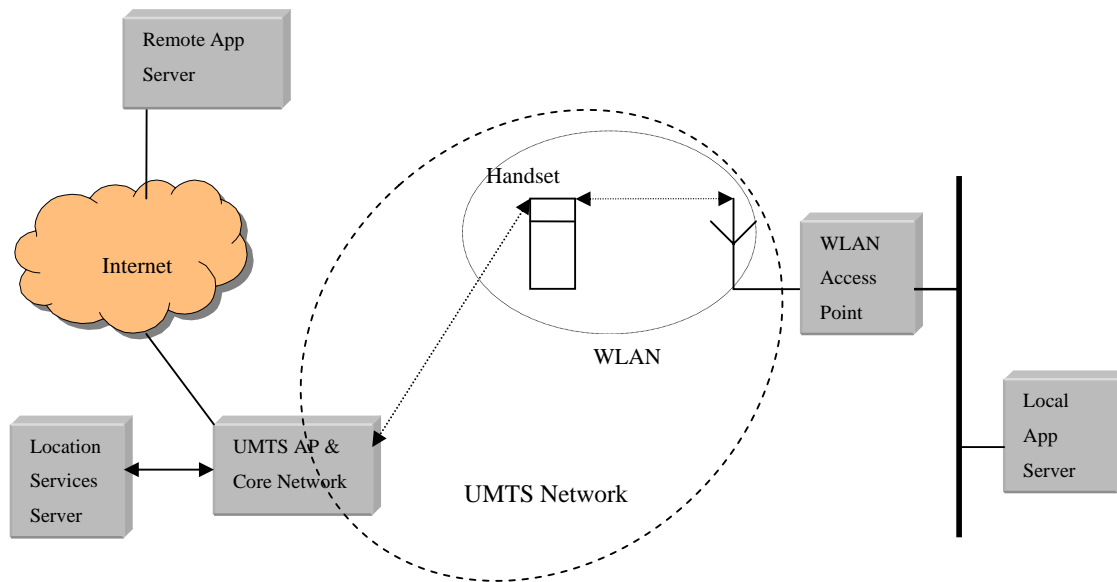


Figure 4.1: Overview of system architecture

In Figure 4.1 a high level overview of the proposed solution to provide unified access to 3G (UMTS) and WLAN based solutions is given. The main elements of the system that we are concerned with are the handset, the local application server which is accessed through the WLAN network and the remote application server which is accessed through the UMTS network. Hanging off each of these servers is a relational database. In order to provide local services in a UMTS network a location based services system is needed. The required location determination is provided by the Location Services Server which hangs off the UMTS core network.

The local WLAN application server is accessed through the WLAN access point using HTTP. The remote application server can be positioned anywhere off the public internet. This is possible as the wireless UMTS and GPRS (2.5G) networks can provide a HTTP connection from mobile client to the public internet – we do not have to get caught up in WAP connections/gateways problems.

4.3 Technologies Used

At the core of this solution is SOAP implementation which gives messaging based communication between mobile client and the two servers over HTTP (amongst other protocols). As stated previously, HTTP can be applied over both the UMTS and 802.11b/g wireless protocols. In this project we assume a standard 802.11b/g wireless network using infrastructure mode and a standard UMTS network.

On the application server runs the SOAP listener servlet which listens for any incoming SOAP messages. This servlet calls the developed Java service methods specified by the SOAP messages and returns the results back to the client. In this system the methods implement database queries for some local service content using the database implementation of Java data base connectivity classes.

4.3.1 Client Design Considerations

The first design consideration for the client was the language that the application would be programmed in. There were a number of options such as:

- Java 2 Platform, Micro Edition (J2ME) – The advantages of J2ME include the ease of programming, portability of code, safe network delivery and good documentation availability. J2ME programming is very similar to ‘normal’ java programming so there would have been no learning curve here. J2ME/MIDP (Mobile Information Device Profile) is also available on nearly all handsets. The main disadvantage is that J2ME is ‘sandboxed’ – this means that with the J2ME APIs the programmer is unable to access the lower level handset functionality. As the design solution required network connection functionality this drawback proved critical.
- C# (C Sharp) – Smartphones with Microsoft compact framework are programmed using C#. This java like programming language offers the developer a large amount of good documentation on APIs and modern integrated development environments. It also provides APIs to the lower level network connection functionality.
- Symbian C++ - The Symbian operating system as described in chapter 3, provides the developer with large functionality and direct access to most of the device’s capabilities. Although more complex to develop in than the other two choices this is

out-weighted by this advantage. As mentioned the disadvantages are the poor documentation, complex code and steep learning curve for a new comer.

After research into the three options above, the decision was made to develop in Symbian as the required WLAN functionality is present and mobile handsets with the necessary WLAN connection card were available. The second option was to go with the Microsoft Compact Framework environment which was not chosen as wireless LAN APIs are not part of the standard SDK and the availability of compatible devices was not assured.

4.3.2 Server Design Considerations

The server design considerations were straight forward. The system design required application servers which run a SOAP implementation servlet. Apache Tomcat and Apache SOAP were the chosen for the application servers as it's free and available for a wide variety of platforms. The service code was written in Java.

4.4 SOAP

SOAP is a simple and flexible XML-based messaging protocol that can provide interoperability of remote applications. It claims to be “a specification for a *ubiquitous XML distributed computing infrastructure*” [12]. The SOAP specification defines a set of rules for structuring messages that can be used for performing RPC-style (Remote Procedure Call) request-response dialogues. It is not tied to any operating system, programming language or transport protocol and can therefore be used for communication between clients and servers running on entirely different platforms. These characteristics make it ideal for LBS systems where the clients are likely to be running different platforms than the servers. With HTTP being the most popular transport protocol, setting up a SOAP dialogue involves only the initiation of a standard internet connection; lower layer protocols including whether connection uses 802.11g or UMTS can be ignored.

While the overhead of the SOAP messages (size) have not made it very popular with mobile communications, this disadvantage is out weighted by the advantages mentioned and by the increased capacity of the UMTS and WLAN networks. In this project the SOAP client will be implemented in Symbian C++ and the SOAP server in Java.

4.5 Server Discovery

As the mobile client will roam into different wireless hotspots and as the handset could be powered off/on at any time, the LBS server on the WLAN and UMTS networks must be discoverable. On the UMTS side, clients will be tied to the service provider network they are subscribed to so the address of the remote LBS server will be known and can be fixed in the application (client roaming to different GSM networks is ignored in this solution). For WLAN connectivity it's not straight forward – the client will roam in different WLAN hotspots that could have been setup by network operators or independent private companies/individuals. Access point names and client IP addresses have to be learned. This is achieved by the WLAN cards on clients scanning the wireless network channels to present a list of wireless network access points present. When an access point is chosen, the DHCP server running on this access point allocates an IP address for the client (infrastructure mode). In our system we the client LBS application also has to discover the address of the server that will be queried. Methods to perform this discovery include:

- setting the access point name be the name of the application server (DNS included in system),
- using a modified DHCP implementation that would allow the application server address to be returned with client address,
- client receiving the application server details in a SMS or WAP push message. Client would have to initiate this by sending SMS to hotspot specific MSISDN.

Chapter 5 – Implementation and Design

This chapter outlines the main implementation and design issues that arose during the development of the project. It will give an overview of the software architecture of both the client and server without going too deep into the actual code. Detail will be given on the client connection handling and SOAP implementation though as this is relevant to the project goal. The SOAP communication between the two will also be described.

5.1 Hardware

The choices for client handset were restricted as the device had to be UMTS (or GPRS) and 802.11g/b (wireless LAN) compatible. The Nokia 9500 was chosen as it has an integrated WLAN card and it is a 2.5G (GPRS) compatible handset. The fact that it is not a UMTS (3G) handset did not affect the results of the project – the lower level communication protocols are not an issue in this project (HTTP can run over both these packet based communication protocols). This handset runs the Symbian 7.0s operating system and uses the Nokia series 80 platform (The handset actually consists of two separate handsets using series 40 and series 80 platforms, but we are not concerned with the former).

A Unix and a Win32 server were used to host the web services with both running an Apache Tomcat application server. The hardware specifications of each are not of importance here.

5.3 Client Architecture

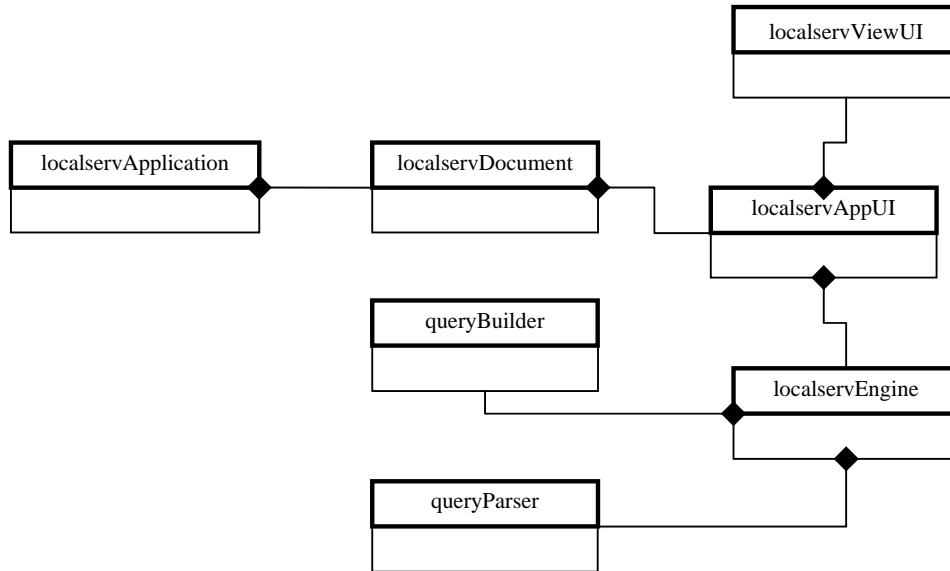


Figure 5.1: Localserv Application Structure

In the above figure (Fig 5.1) the main classes in the Localserv application are given. The standard Symbian application framework and a client engine were developed. All persistent data, like the structs `iQuery` and `iResult` are created in the Document. The `localservEngine` manages all connection functionality and calls both the `localservQueryParser` and `localservQueryBuilder` to generate the SOAP messages.

The `localservAppUI` controls the user actions functionality e.g user selects a menu item. It also creates `localservAppView` which displays the UI on screen.

5.3.1 Client Connection handling

The client application was required to initiate a HTTP connection over the GSM and WLAN networks. To simplify this, and not to get involved in lower level socket issues, the standard Symbian access point functionality was taken advantage of. This did add some constraints to the application but did not affect the overall goal.

In standard Symbian devices the network interfaces are defined by Internet Access Point (IAP). An IAP defines the network ID, type, access point and other connection details, and

is used for all network connections. These IAPs are user created or pre-provisioned on the handset by network operator. In normal operation, when establishing a data connection to a network, the user is prompted to choose which access point to use. In Localserv the IAPs to use for the internet connections were hard coded. This allowed the application to connect to both the WLAN and GPRS networks without user intervention - but does require that the IAPs are setup with known fixed names beforehand. The difficulties with this approach was that the standard simple connection functionality had to be changed and more complex code written to get the access point name and ID from the Comms Table. A new connection method was also needed.

5.3.2 SOAP on client

Unfortunately support for SOAP in the Symbian OS is non existent. This required an implementation which would use standard SOAP to be developed from scratch. To add to this task there is also no XML API (In standard Symbian documentation an XML parser/creator is documented but on closer inspection it was found that it's not shipped with the current SDKs at all!). The XML task was made easier as a freely available Symbian implementation called SyExpat was found [9]. SyExpat offers a XML reader and generator for Symbian. This was ported over to the series 80 SDK environment and built successfully.

Using SyExpat two classes were developed call `loacalservQueryBuilder` and `localservQueryParser` to handle the SOAP message generation and parsing. These classes were modified standard XML generator and parser classes with SOAP elements added using string constants. This functionality was not flexible and required a lot of text parsing and generating code that would not be needed in if a SOAP API was present. It also restricted the requests and replies that could be generated but was sufficient for this project.

5.4 Server Architecture

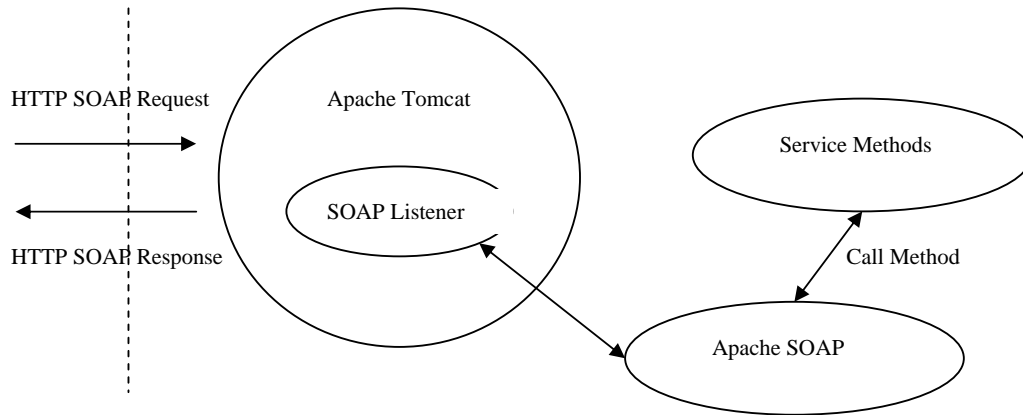


Figure 5.2: Server Architecture

The architecture of the server (Localserver) is given in Figure 5.2. It is a standard Apache SOAP implementation with the following service methods developed in Java:

```
getServiceProviderInfo(int hostID,string location);  
getServiceList(int serviceProviderID);  
getServiceObject(int serviceListID);
```

For this project the data returned was fixed i.e. no database querying was performed. Both the local and the remote application servers ran the same code.

5.5 SOAP Implementation Details

The goal here was to design an efficient SOAP implementation that could return different types of service objects and could also run on a Symbian Client. The following is an overview of the design used to apply the location based services in a system using both WLAN and UMTS networks. The logic flow is as follows:

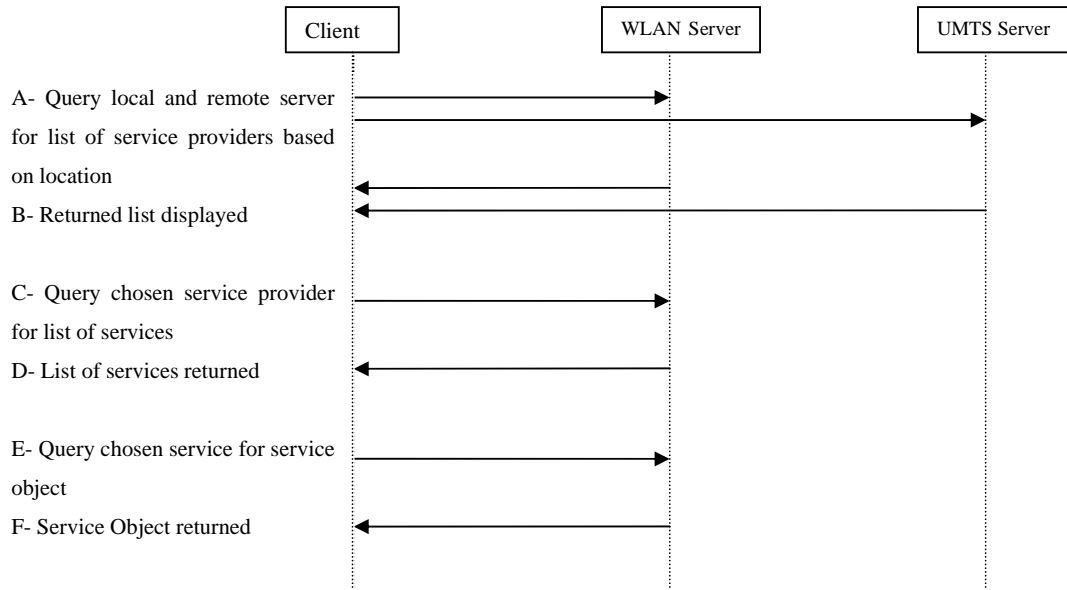


Fig 5.3: SOAP dialogue

Below is the user defined data type and the Java method prototypes used in SOAP engine (called LocalServServer) on the server. There is no difference between the queries to the WLAN and the UMTS servers; the responses are the identical as well - as far as the SOAP layer is concerned its irrelevant. This simplifies the client SOAP implementation and allows duplication of server code. To allow the user to be informed if the service is on the WLAN or UMTS server a flag is set in the info string in response B.

Complex Data Type:

```

ContentDetails {
    int contentID;
    string info;
    string label;
}
  
```

Prototypes:

```

ContentDetails[] getServiceProviderInfo(int hostID, string location);
ContentDetails[] getServiceList(int serviceProviderID);
ContentDetails[] getServiceObject(int serviceListID);
  
```

A key idea of this design is the use of numeric identifiers for all content. The LocalServServer would retrieve from a relational database the data for the three prototypes above. The data in this database would be structured such that every element would have a

numeric identifier. These elements could be service objects or lists containing the identifiers of other elements and the text to be displayed.

To illustrate this functionality the example of the local restaurant menu will be used; User in location (x,y) starts the Localserv application to find a local restaurants menu list:

- Localserv application starts and generates SOAP message using with location (x,y) to query known WLAN and UMTS server for list of service providers (Step A in fig. 5.1). SOAP message decoded on servers and `getServiceProviderInfo` method called.
- This method returns an array of type `ContentDetails` where the `contentID` references a service provider and the text string gives its name.
- When the client selects a particular service provider, the SOAP query messages specifies the `contentID` which is used to get the list of services for that service provider from the database.
- The same idea is repeated to get the service objects.

In this project the return data is fixed and there is no database, however the fixed data structure does replicate what would be returned from a database.

Chapter 6 – Results Analysis

6.1 Results and design analysis

The Localserv application running on the Nokia 9500 mobile handset successfully queried a local application server over the wireless LAN and a remote server over the GPRS network using SOAP queries. The returned local services were displayed by the client. The SOAP messaging using HTTP over these networks was straight forward to implement as the wireless network protocol underneath could be ignored. This simplified the application a great deal. The overhead of the SOAP messaging was not considered here as the system was too simple.

The code that was added to allow connection to both network proved very troublesome to get working. Using the predefined/fixed access points and application server address simplified application as none of these parameters had to be discovered. As it was a closed development system with no unknowns this worked well and did not affect the proof of the concept. The network scanning and server discovery functionality could be added in the future to prove concept in real life systems with unregulated hotspots. Even though the location measurement was not calculated by the application how it would be used was illustrated. The reason for using fixed values is that the network functionality to calculate user position could not be accessed.

The development of the Localserv client application in Symbian C++ was unexpectedly difficult and consumed a lot of time. Not having SOAP or XML APIs in the standard SDK added to this. The source code for the Localserv application consisted of over two thousand lines of code for what is a relatively simple application. The UI presentation of the data was very basic and sufficed for this ‘proof of concept’ project. Difficulties with the view architecture and the dialog framework were the reasons for this.

The design of the SOAP messaging framework was a success and provides the bases for further work here. Using identifiers for all service content, be it lists or actual objects, simplified the queries and allowed the same SOAP message structure to be used for all queries and responses. This helped on the client side SOAP implementation that had to be coded from XML parsing APIs and will work well if content put into relational database

Chapter 7 – Conclusions and Future Work

7.1 Future Work

It is hoped that this project provides a good bases for the future development of a complete client and server side solution that provides unified access to local services. The identified future tasks can be split into three areas: client development, server development and network functionality.

Client Development

The concepts of how the local application server is discovered in WLAN hotspots needs to be investigated. This is core to a system like this working in a real environment. Future projects should also look at implementing the client on different operating systems like the Microsoft Compact Framework and Java. In the current Symbian application, development work includes: a better UI, more SOAP messaging functionality and better access point selection code.

Server Development

The server application the addition of a relational database and an intelligent database design has been highlighted as a future task. With this more complex SOAP messaging and service methods would be needed. Good database design and SOAP servlet development would be required.

Network Functionality

Addition of a location estimation system that could return the users estimated position. Different location determination techniques and their accuracy could be investigated. Also, how Parlay API are implemented, how they could be used and what use can be made of the core network functionality offered.

7.2 Conclusions

The goal of developing an architecture that allowed unified access to WLAM and 3G local services was successfully demonstrated. The client and the server designs worked well. A number of limitations, like no real application discovery in the WLAN network, were introduced but I don't think they affect the proof of the concept of the project. The overall software architecture worked well but unfortunately more functionality was not developed.

The decision to develop in Symbian C++ proved to have a huge influence on the project. This development environment was very difficult and slow to learn and continually frustrated after that – doing the simplest things in Symbian are never easy. As a development environment for demonstrating a concept, like we did here, it is not the ideal. Too much time was spent getting simple functionality working rather than on the concept itself. Poor development IDE and documentation resulted in the Symbian learning process taking up the largest amount of time in this project.

The project proved to be a challenging experience and a great deal on location based services, Symbian development and SOAP was learned.

References

- [1] William Lehr and Lee W. McKnight ,”Wireless Internet Access: 3G vs. WiFi?”, August 23, 2002
- [2] M. Spanoudakis, A. Batistakis, I. Priggouris, A. Ioannidis, S. Hadjiefthymiades, L. Merakos, “Extensible Platform for Location Based Services Provisioning”, *Proc 4th International Conference on Web Information System Engineering Workshops*, July 2004.
- [3] <http://www.3g.co.uk/PR/Oct2003/5950.htm>
- [4] D. Mohapatra and Suma S.B., “Survey Of Location Based Wireless Service”, IEEE ICPWC, 2005
- [5] M Tayal, “Location Services in the GSM and UMTS Network”, IEEE 2005
- [6] R-H Jan and Y-R Lee, “An Indoor Geolocation System for Wireless LANs”, *Proc 2003 International Conference on Parallel Processing Workshops*, 2003.
- [7] J.W. Hellenthal, F.J.M. Panken, M. Wegdam, “Validation of the Parlay API through prototyping”, IEEE 2001
- [8] Leigh Edwards and Richard Barker, “Developing Series 60 Application”, Addison Wesley, 2004- ISBN: 0-321-22722-0
- [9] Christoffer Andersson, “GPRS and 3G Wireless Applications”, Wiley, 2001 - ISBN: 0-471-41405-0
- [10] http://www.simkin.co.uk/Demo_Cpp_Symbian.shtml
- [11] <http://www.w3.org/TR/SOAP/>
- [12] Steve Graham, Simeon Simeonov, Toufic Boubez, Doug Davis, Glen Daniels, Yuichi Nakamura, Ryo Neyama, “Building web services with Java : Making Sense of XML, SOAP, WSDL, and UDDI”. - Sams Publishing, 2001. - M0098634DC

Appendix 1

You should put into an appendix, material which is required to be present with your dissertation, but which would interrupt the flow of the text if presented in the main body of the document.

Glossary

UI	User Interface
LBS	Location based Services
GPRS	General Packet Radio System
UMTS	Universal Mobile Telephone System
OMA	The Open Mobile Alliance
MLP	Mobile Location Protocol
IETF	Internet Engineering Task Force
OGC	Open GIS Consortium
OTDOA	Observed Time Difference on Arrival
EOTD	Estimated Observed Time Difference
CI	Cell Identifier
TA	Time of Arrival
IAP	Internet Access Points