Mobile Control System
for Location Based Alarm Activation

Jan Magne Tjensvold

June 16, 2008

Abstract

This report describes the design and implementation of a system that can automatically control various services based on the location of one or more mobile devices. These services can also be controlled manually through a user interface on the mobile devices. A burglar alarm service that can automatically be activated and deactivated is used as a case study for this system. The implementation is entirely Java based, using the Android operating system to run the mobile device software. Challenges related to accurately locating the mobile devices and communicating between the mobile devices and a home server is examined. A set of policies for activation and deactivation of the alarm system and other services is also defined. The report also looks at examples of other services like automated temperature, lighting control and adaptive fire sensors that can be integrated into the same system.

Acknowledgments

I wish to thank Hein Meling for his detailed and insightful comments on the report and his helpful ideas on the design and implementation of the software. Also many thanks to Thanh Danh Nguyen for his useful information regarding fire alarm systems.

Contents

Chapter 1

Introduction

Networked computer systems have become prevalent in most aspects of modern society, and we have become dependent on such computer systems to perform many critical tasks, such as health monitoring and alarm notification. Wireless and ad-hoc communications technologies enable the elimination of physical cables between residential home network entities, such as computers, set-top-boxes (STB), sensors and control units, and are typically less costly to install than their wired counterparts due to cabling. These technologies have opened up a whole range of new applications in the utility segment for residential homes, such as automatic meter reading, remote control of heating, and security and safety systems. However, interacting with such systems is still somewhat cumbersome and inflexible.

This report presents a mobile control system (MCS) that can control various services based on the location of one or more mobile devices. A service for activation and deactivation of a residential burglar alarm system is used to demonstrate the feasibility of this system.

The report looks at the design and implementation of a system that can detect when a user *enters* and *exits* a specific area. Based on the location of the user the system can automatically control a set of services running on a *home server*. By carrying a *mobile device* the user can also interact with these services through a user interface. The home server software will also be able to track the presence of multiple mobile devices.

Figure 1.1 shows the main components of the MCS described in this report. The main challenges are how the home area is defined, how the mobile device is able to recognize this area and what should happen when an individual mobile device enters or leaves the area. The goal of the system is to automatically control a burglar alarm system in the house where the home server is located. Localization/positioning, flow of event notifications and the home server logic has been the main focus. Implementation of the low-level sensor communication and electrical wiring needed to interface with an actual burglar alarm system is not considered in this report. Instead the alarm system and associated sensors are simulated.

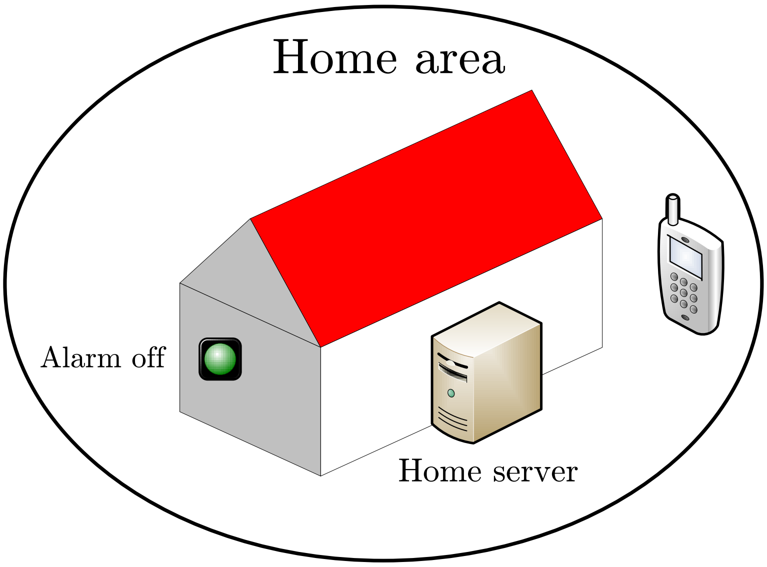


Figure 1.1: Main components in the Mobile Control System

The alarm system scenario helps to demonstrate the functionality of the system, but in principle the MCS is not restricted to only controlling alarm systems. With a bit of work it is able to control any number of different systems as long as the enter/exit area events can be translated to useful actions for the specific system. In terms of the alarm system, the enter/exit events are mapped to alarm deactivation/activation.

The system will try to find a way to deactivate the alarm if a user enters the home area while the alarm is activated. This might involve asking the user to enter a PIN code before the actual deactivation takes place. In the opposite direction the system will try to find a way to activate the alarm when the last user exits the home area. In this case the system can decide to automatically activate the alarm or ask the user first.

The term "home area" is meant to describe an area that covers a specific point of interest, like a house, apartment or some other kind of building. Unless explicitly stated otherwise, the rest of this report uses the term in this general sense. In practice a home area can consist of multiple smaller overlapping and/or adjacent areas. An example would be the home area of a large building with multiple wireless networks installed. In this case the home area could be defined as the combined coverage area of all the wireless networks. Only when a user was not connected to any of the networks would the system consider him to be outside the home area.

One important assumption for this system is that the location of the mobile device should reflect the location of the user whenever he enters or exits the area specified by the system. If the user forgets to bring his mobile phone with him when he leaves the house, the alarm will not be automatically activated. For this reason the system is designed as a supplement to the regular wall mounted keypad used by home security systems. This allows the alarm to be activated and deactivated by conventional means if the user forgets or loses his mobile phone.

By connecting the MCS to a central heating system the enter/exit events could be mapped to automatically adjust the temperature up/down to reduce energy consumption when nobody is home. This can also be applied to central lighting control systems so that the lights in a house can be completely turned off when the house is empty. When everyone has left the house, advanced systems that monitor the electrical equipment in a house could automatically turn off equipment that poses a fire hazard. Intelligent fire alarm systems could also be implemented. The sensitivity of the optical fire detectors can be automatically increased when the house is empty. Depending on the capabilities of the detector this would allow the fire alarm system to determine if someone left their lit candles unattended. In this case all the users would be notified (instead of triggering a full fire alarm) and the fire alarm system could turn the sensitivity back to normal to check that no real fire has started.

Software has been implemented on both the home server (MCS Control Server) and for the mobile device (MCS Droid) to enable them to coordinate their efforts. Depending on the capabilities of the mobile device Bluetooth, WiFi or GPS can be used to detect when the user enters or exits the home area. For Bluetooth and WiFi the home area is defined by the coverage area of the network they provide. For GPS the home area can be defined in any number of ways, but for the sake of simplicity the implementation of MCS only uses a circular geographical area.

The communication between the mobile device and the home server can take place across any available Internet access services, such as GPRS or even WiFi, as long as the technology is supported by the mobile device. Figure 1.2 shows the home server located behind a firewall and how it is able to communicate with the mobile device through the infrastructure of a GSM or a WiFi network.

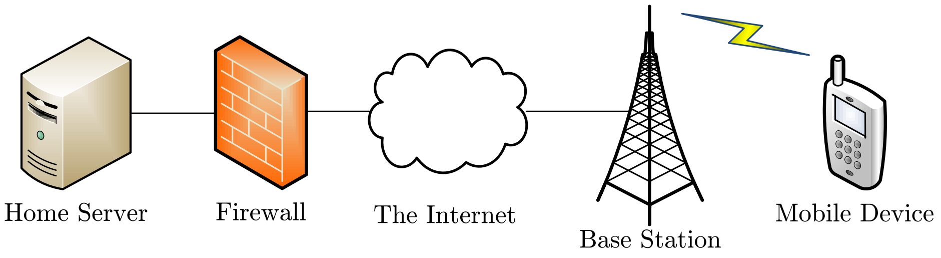


Figure 1.2: Physical connection between the home server and mobile device

1.1 Related work

There exists a number of location/proximity-based systems that can automate simple tasks such as locking/unlocking your computer and launching various applications by using a Bluetooth enabled computer and mobile phone. Examples of such systems includes floAt's Mobile Agent (Windows based for Sony Ericsson handsets) [5], metaquark Home Zone (Mac OS X) [8] and BlueProximity (Linux) [1].

In the home automation area there are several systems that allows users to control just about every piece of electronics, including security and surveillance systems. However, most of these systems are not location-based and due to the number of different devices they can control they often have relatively complex control interfaces. NorAlarm has developed a mobile application that enables users to control an alarm system through a Java ME user interface [43]. This application allows an alarm system to be manually controlled, but it does not include automatic activation and deactivation based on the location of the user.

1.2 Report organization

This report is organized into several chapters. Chapter 2 examines some of the mobile application platforms currently available and different techniques to determine the whereabouts of the mobile devices. Chapter 3 discusses different aspects of the design and implementation. Chapter 4 describes some test and simulation results and Chapter 5 concludes this report.

Chapter 2

Background

2.1 Mobile application platforms

Platforms for developing applications for mobile devices (mobile phones, smart phones, PDAs, etc.) exist in many different shapes and sizes. Some of them are entire integrated operating systems while others provide only a minimal set of APIs to create interactive multimedia applications. This section gives a background description of the some of the popular alternatives available on the market.

2.1.1 Android

Android [27] is the Open Handset Alliance's (OHA) mobile operating system. It is based on the Linux kernel and the application platform is very similar to Java SE. Android uses Apache Harmony's [22] class library where only a few of the original Java SE packages have been removed. This mainly includes the Java SE Swing and AWT GUI packages. These have been replaced by GUI packages that are more suited for the reduced screen sizes used by mobile devices.

Android uses a special Java virtual machine (JVM) called Dalvik which is optimized for running in multiple instances so that each process can have its own JVM. Instead of Java class files the Dalvik JVM executes files in the Dalvik Executable format (dex files). dex files are in reality Java class files that have been preprocessed to reduce the amount of memory they use.

The Android platform is currently in beta and is lacking official documentation on some of the APIs. This includes the WiFi and Bluetooth APIs which are present in the Android class library, but not fully completed. Android is a relatively new platform and currently no mobile devices running Android exist. It is mainly aimed at medium and high-end mobile devices featuring GPS and hardware 3D graphics acceleration. The first real Android based phones are expected to be released in the last half of 2008 simultaneously with Android version 1.0.

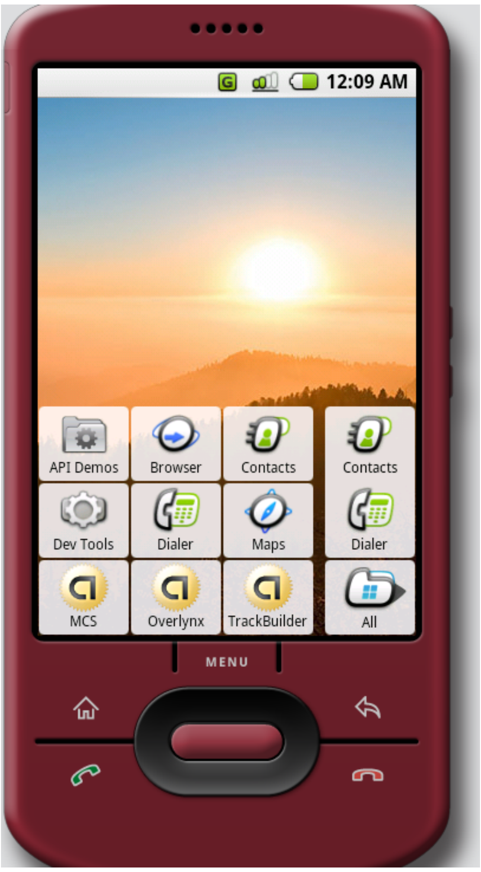


Figure 2.1: Screenshot of the Android Emulator

The Android SDK is available for Windows, Linux and Mac OS X, free of charge. Developers can use popular Java development tools like Eclipse and Ant. Existing Java SE based code can also be ported to Android with relative ease, as long as it does not interface with any of the packages that have been removed. The SDK also includes an emulator as shown in Figure 2.1 to allow most aspects of an application to be tested before the real devices are released. The premise of the Android platform is to be a fully open and free mobile platform. Much of the Android platform has already been released under an open source license. It is expected that Google, the main OHA developer behind Android, will release the remaining parts as open source in time for the release of version 1.0 or soon thereafter.

2.1.2 Java ME

Java Platform, Micro Edition (Java ME) [36] is, according to Sun Microsystems, “the Most Ubiquitous Application Platform for Mobile Devices”. It is arguably one of the most widely available mobile application platforms and as many as 8 out of 10 phones ship with Java ME [40]. Java ME, similar to its big brothers Java SE and EE, does not impose any elaborate certification, approval or other costly programs upon developers so that they can develop and release their work.

Java ME consists of a set of configurations and profiles. Of all the classes and packages in the Java SE API, only the java.io, java.lang and java.util packages remain in the standard Java ME environment. The only remaining parts of the collections API is the Hashtable, Stack and Vector classes. The latest version of Java ME does not even guarantee support for floating point numbers, but this is expected to be corrected in the next version. Mobile application extends the MIDlet class which takes care of the application lifecycle. There are also basic frameworks for making network connections and saving data to persistent storage.

Java ME runs on several different mobile operating systems from a wide variety of handset manufacturers. Each manufacturer usually provides a Java ME SDK with their own version of the Java ME emulator provided by Sun Microsystems. Because the number of APIs available in the standardized profiles and configurations are relatively limited, manufacturers usually provide a set of extension APIs. Java ME also specifies a set of optional packages for things like Bluetooth communication, location-based services and 3D graphics. The main challenge developing for this platform is that many of the APIs often are optional and some are vendor-specific. This often breaks the Java concept of "write once, run anywhere" as one has to account for unavailable APIs and different APIs that accomplish similar tasks. Sometimes the implementation of Java ME might differ from vendor to vendor making it even more difficult to write applications that will run on all target devices.

Java ME SDKs are available for both the Windows and Linux platforms. However, there are no officially supported SDKs that allows developers to work on the Mac OS X platform. Sun is currently working on a mobile operating system called JavaFX Mobile [38] which will be able to run Java ME as well as Java SE applications. It will mainly be aimed at high-end mobile devices and the hope is that the Java SE support will prevent some of the trouble inherited by Java ME.

2.1.3 iPhone

The iPhone [14] is a high-end mobile phone developed by Apple. It was first introduced in the middle of 2007 and it runs the iPhone OS which is developed specifically for the iPhone hardware. Version 3.1 and later of Xcode, which is Apple's suite for developing software for Mac OS X, is also used to develop software for the iPhone. Xcode can be downloaded free of charge and is currently available for Mac OS X v10.5 (Leopard) only. iPhone software is mainly developed using the Objective-C language which is also commonly used by Mac OS X developers.

To develop software for the iPhone platform is free, as long as you have a computer running the correct version of Mac OS X. The iPhone SDK includes an emulator which enables one to develop software without the need to own an actual iPhone. However, in order to run your software on a real iPhone device or to release it requires membership in the iPhone Developer Program. Currently you must be at least 18 years old and resident of the USA to be able to apply for this program. The program is currently in beta and Apple is only accepting a limited number of developers into the program. The iPhone Developer Program comes in a standard and an enterprise version, both of which costs money.

To be accepted into the developer program one must first accept Apple's Registered iPhone Developer Agreement [15]. This agreement could potentially conflict with many free software licenses like the GPL [56, 26] which could make it difficult to write free software for the iPhone, unless one is able to acquire special permission from Apple. Before an application can be publicly released it must first be approved by Apple. Upon successful approval, Apple cryptographically signs the application so that it can run on any iPhone. The only legally sanctioned arena for distribution of iPhone applications is the Apple iTunes Store.

Many of the technical restrictions imposed by Apple can be circumvented on individual phones by a process aptly called jailbreaking. However, this may very well void the warranty on your iPhone and in the worst case cause the phone to stop functioning.

On the technical side, the iPhone platform provides access to many APIs which are similar to those in Android, including APIs for Location-based services and OpenGL for Embedded Systems (OpenGL ES). The Bluetooth API for the iPhone is not documented and the Bluetooth capabilities of the device is less than fully featured.

2.1.4 Windows Mobile

Windows Mobile is based on Windows CE and is Microsoft's proprietary mobile operating system. Visual Studio is the main development tool used to develop applications for the Windows Mobile platform. The .NET Compact Framework allows developers to write applications using a subset of the full .NET Framework used for desktop development. It is also possible to write unmanaged code in C or C++ for the platform. The Windows Mobile SDK contains an emulator like most of the other popular mobile application platforms. There is no location provider independent location API, but serial COM port programming against a GPS is possible. Windows Mobile also has an API for extracting Cell-ID information and a WiFi API that provides signal strength and other useful information.

2.1.5 Other platforms

In addition to the platforms for mobile application development described above there exists a large section of less popular alternatives. Symbian OS [48] is Nokia's primary mobile operating system. Symbian's main development languages is C++, but many Symbian devices can also be programmed in additional languages such as Python, Perl, Visual Basic and Java ME. C++ development for the Symbian platform has been widely regarded as difficult because of the many complicated low-level techniques needed to make applications function.

Motorola's Linux based mobile operating system is called MOTOMAGX [41]. The ACCESS Linux Platform (ALP) [18] is a fully fledged Linux based operating system that can run Java ME, Garnet OS (formerly Palm OS) and native Linux applications. There is also a game-oriented platform called Mophun [16] that is available on some Symbian based handsets. Adobe Flash Lite [49] is a light weight version of Flash that allows developers to make multimedia applications. There also exists bridges for Flash Light that allows it to access much of the same functionality as Java ME [20]. Research In Motion's (RIM) BlackBerry platform was previously programmed in C, but has moved to a Java ME based system with BlackBerry specific extensions. Trolltech has an application platform called Qtopia [52] which is based on Qt.

2.1.6 Summary

Figure 2.1 shows a comparison of the five most popular mobile application platforms. Android is the application platform that was chosen for the implementation of the mobile client software. The combination of Android's free and open source initiative, the Java SE like set of APIs, availability of location-based service APIs and Android's event-based architecture made Android the number one choice. When the real mobile devices arrive Android will hopefully deliver on its promise to be a fully open and free mobile platform.

	Language	Generic Location API	Cost of Development	Code Signing	Available Hardware
Android	Java	Yes	Free	None	None
Java ME	Java	Optional[1]	Free	Optional	Very High
iPhone	Objective-C	No	Min. $99	Required	Medium
WinMob.	.NET[2]	No	Free[3]	Optional	Medium

[1] JSR 179: Location API for J2ME [39].

[2] Supports managed (C# and Visual Basic) and unmanaged (C++) .NET.

[3] Programs can be written with the basic SDK and compiler using a simple editor, but Visual Studio itself is not free.

Table 2.1: Comparison of different mobile application platforms

2.2 Localization

Localization or positioning is the process of determining the whereabouts of a specific entity. In our case we wish to determine the location of one or more mobile devices. A key component in location-aware systems, such as the MCS, are services that can provide information about where the user is located. Services that harness this information are commonly known as location-based services (LBS) [47].

Different kinds of localization techniques can be applied depending on the hardware present in a mobile device. The ideal technology would provide unlimited accuracy for every conceivable location on the entire planet. However, no such silver bullet technology currently exist, nor is it likely to be available in the foreseeable future.

As always some trade offs exist between the different techniques available. For mobile devices the important characteristics are:

- Energy efficiency for long battery life [54],
- accuracy of the location information,
- performance in different environments and
- availability on the market.

Power saving strategies such as periodically turning on/off the location provider services can increase the energy efficiency and battery life. To provide the most energy efficient and best possible accuracy the mobile software will be able to make use of any of the techniques described below.

An example of how a location provider can be selected in Android and Java ME is by specifying a set of criteria[1]. This allows an application to find the best possible location provider instead of relying on the existence of one specific location provider. The method that finds the best location provider evaluates the following criteria:

- Desired maximum power consumption (low, medium or high),
- accuracy (in meters),
- support for altitude measurement,
- support for bearing information,
- support for speed information and
- if the location provider cost money to use.

The following sections looks at some of the most common technologies available and assess their strengths and weaknesses.

[1] The design of the Android and Java ME location APIs are very similar and they use almost the same criteria to select location providers.

2.2.1 GPS

Global Positioning System (GPS) [45] is a satellite based, medium earth orbit (MEO), navigation technology. It is used by both civilians and in military operation. GPS relies on a constellation of at least 24 satellites to provide location, speed and direction information to its users. It works by using a technique called trilateration combined with atomic clocks in the satellites in order to accurately determine the correct location.

As of 2006 only 11.1% of mobile phones have an integrated GPS receiver [33] which makes this a relatively low availability technology for our target devices. The accuracy of GPS is relatively high compared to most other techniques, but it requires line of sight to satellites which severely limits its use indoors. In big cities with lots of high buildings and narrow streets GPS will often have very low accuracy because the number of satellites it can see is limited.

Because GPS provides high accuracy geographical coordinates the MCS home area can in principle be defined as any irregularly shaped area. For the sake of simplicity a circular area is often used as that greatly simplifies the algorithm required to find out if a user is inside or outside the area.

A disadvantage of GPS is the delay it takes from the time when the GPS is enabled until it can deliver the first location information (positional fix). Warm start (GPS has been off a few hours) usually gives a positional fix within 10 – 20 seconds, while a cold start (some of the satellite data is outdated) can cause the positional fix to take up to several minutes. The power consumption of high-end GPS devices has been found to be surprisingly low. Ultra-low power GPS chips targeted at mobile devices, like the GloNav GNR1040 RFIC and Broadcom BCM4750, can use as little as 15 mW.

2.2.2 WiFi

WiFi (also known as WLAN) is a low tier, terrestrial, network technology for data communication. The WiFi standards operates on the 2.4 GHz and 5.8 GHz industrial, scientific and medical (ISM) frequency bands. It is specified by the IEEE 802.11 standard [29] and it comes in many different variations like IEEE 802.11a/b/g/n. The application of WiFi has been most visible in the consumer market where most portable computers support at least one of the variations.

The coverage area of a local WiFi network can be used to determine if a mobile device is inside or outside the home area. The mobile device will store the network identifier (access point MAC address) so that it later can recognize the WiFi network when it enters its coverage area. There are some services that can map wireless network identifiers to geographical locations such as Skyhook Wireless [10] and WiGLE [11], but their coverage varies widely depending on where in the world you are located. A disadvantage of using a WiFi network is that it never produces a perfectly circular deterministic coverage area. Due to environmental conditions like physical obstructions and radio interference the coverage area may even vary over time. The indoor range of WiFi is around 20 – 50 meters (70 meters for 802.11n) depending on the environment and which standard is used. The power consumption of WiFi chips, like the 54 Mbps capable Broadcom BCM4326 and BCM4328, is less than 300 mW. This puts the power consumption of WiFi about 20 times that of a GPS in mobile devices.

2.2.3 Bluetooth

Bluetooth is a low tier, ad hoc, terrestrial, wireless standard for short range communication. The IEEE 802.15.1 standard [30] forms the basis for the Bluetooth wireless communication technology. It is designed for small and low cost devices with low power consumption. The technology operates with three different classes of devices: Class 1, class 2 and class 3 where the range with a clear line of sight is about 100 meters, 10 meters and 1 meter respectively. Wireless LAN operates in the same 2.4 GHz frequency band as Bluetooth, but the two technologies use two different signaling methods which should prevent interference.

Similar to WiFi in many ways, the most common Bluetooth devices are class 2 and have only a range of about 10 meters. A distinct advantage of Bluetooth over WiFi is that it has a power consumption in the same range as a GPS receiver. Its short range allows for confined location areas that can be used to detect when the user approaches the main entrance or the garage door.

2.2.4 Cell-ID

Base stations in cellular phone networks, known as base transceiver stations (BTS), like GSM and UMTS emit unique IDs which can be used to determine the location in a similar manner as WiFi. This means that Cell-ID is available in just about every mobile phone in existence. Although Cell-ID by itself is not a good enough location technique for most LBS [51] combined with other techniques it can still be very useful. The ubiquitous nature of Cell-ID has resulted in a large amount research into techniques to improve its accuracy [17, 32, 35, 46, 53, 55].

The accuracy of Cell-ID is highly dependent on the cell size, which causes the accuracy to vary from a few hundreds meters to several kilometers. Areas with a high population density requires a high density of BTSs (smaller cells) to serve a larger amount of customers. This results in the highest accuracy for urban areas and gradually reduced accuracy for suburban and rural areas.

To uniquely identify a cell its Global Cell Identity (GCI) contains information about the country, mobile network operator and the location area where the cell is located. Currently no known network operators provide publicly available directory services that can resolve Cell-ID information into geographical coordinates. However, efforts like CellDB [2], CellSpotting [3], CellTrack [4], GSMLoc [6], OpenCellID [13] and Intel's Place Lab [19] provides access to databases that are able to map some Cell-IDs to the approximate latitude and longitude of the associated BTSs. Google Maps with My Location (GMML) [28] builds on a similar approach by collecting Cell-ID information from users to improve accuracy. Navizon [9] has also assembled a database with WiFi and Cell-ID information. However, GMML currently does not provide a public API so that others can access their data, and Navizon only provides Cell-ID positioning (not WiFi) with their free SDK for non-commercial use.

As a last resort the Cell-ID information can be sent to Yahoo!'s [57] ZoneTag Cell Location API which maps it to an address. This address can then be fed into the Yahoo!'s Geocoding API which resolves the address to a latitude and longitude. These APIs will do their best to find the location of the base station, but often the resulting coordinates are way off course. In a worst case scenario ZoneTag is only able to find the country where the cell is located and Geocoding will provide coordinates pointing to somewhere in the middle of the country.

An advantage over WiFi location lookup services is that the quality of the data in Cell-ID directories remain relatively constant over longer periods of time. The rate of new WiFi networks appearing and old ones disappearing is much higher than that of BTSs because of the large difference in cost between the two. This causes the quality of the collected data in WiFi location directories to degrade much more quickly over time, unless extraordinary measures are taken to actively update and maintain the data.

2.2.5 Summary

Table 2.2 shows a comparison of the four location provider techniques. Very high accuracy is not a necessity for the application considered in this report. WiFi and GPS are suitable location providers, but Cell-ID and Bluetooth can be used as a supplement. The growth rate of GPS enabled mobile phones is expected to increase as LBS becomes an integral part of the mobile experience. Together with other techniques this will enable ubiquitous access to high quality location information.

	Power	Range			Available	API
	Consumption	Indoor	Outdoor	Accuracy	Hardware	Support[1]
GPS	Low	N/A	N/A	Very high	11.1%	Medium
WiFi	High	Low	Medium	Medium	70%[2]	Low
Bluetooth	Low	Very low	Low	Medium	90%[2]	Medium
Cell-ID	Very low	High	Very high	Very low	100%	Medium

[1] Based on platforms supported and consistency of APIs across different hardware

[2] Guesstimate

Table 2.2: Comparison of different location provider techniques

Chapter 3

Design

The MCS has been implemented in Java and the source code is released under the open source Apache License, Version 2.0 [24]. See the project web page at `http://projects.ux.uis.no/projects/show/mcs` for instructions on how to access the source code.

Figure 3.1 shows the main use cases for the mobile client user. These are the actions that a user could reasonably expect to be able to perform from the mobile device client software. The main focus of this report is the policy configuration, home area configuration and the enter/exit home area use cases.

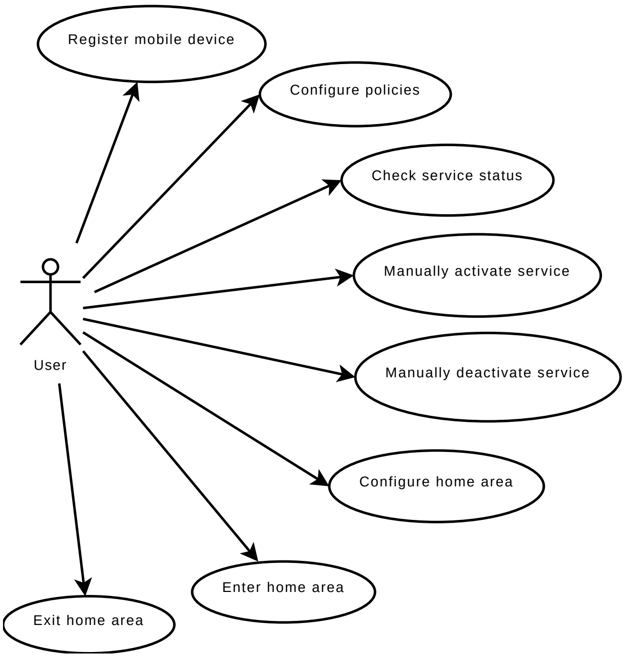


Figure 3.1: Use case diagram for the mobile client user

3.1 Overview

We begin by giving an overview of the components that together make up the mobile control system (MCS). The MCS consists of the software on the home server and on the individual mobile devices. The most important part of the software is the two main applications:

Control server is the main control application that runs on the home server. It manages the set of associated mobile devices and the available services.

Droid is the mobile application that runs on Android based mobile devices. It monitors the location of the mobile device and notifies the control server when the user changes location. It also displays notifications and requests from the control server and allows users to manually control some of the services.

In addition to the two main applications a couple of auxiliary applications were implemented to test the system:

Alarm server is an application that emulates a burglar alarm system on the home server. It enables users to activate, deactivate and query the status of the alarm system.

WiFi simulation server gathers information about the WiFi connection of the device it is running on. Droids can then connect to this server and use the received data to emulate the presence of an actual WiFi API. This is necessary because the WiFi API in Android is unavailable at the present time.

Finally the implementation contains a set of utility modules:

Core is a shared set of utility and base classes for both mobile and home server applications.

JMS common contains utility classes specifically created for JMS communication.

WiFi simulation common contains a class that specifies the protocol used by the WiFi simulator.

Apache Maven 2 [23] is used as a tool to manage the external project dependencies and the build process. Figure 3.2 shows the dependencies between the Maven modules that the implementation is divided into. These modules correspond to the seven parts of the system described above.

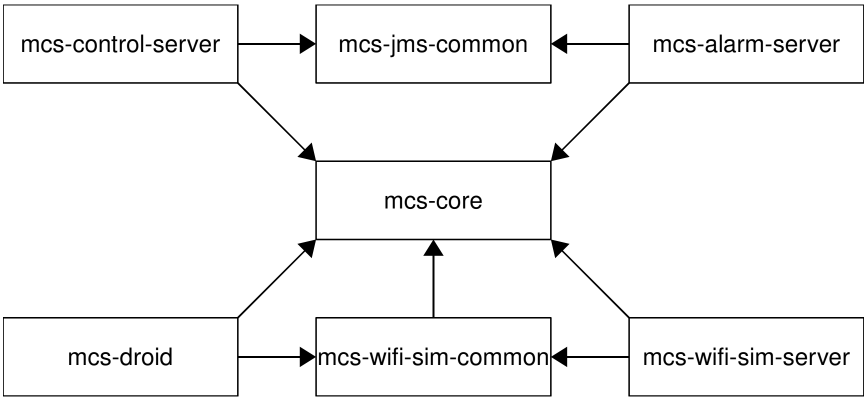


Figure 3.2: Software module dependencies

3.2 Architecture

To build a system with the complexity of the MCS an architecture first had to be established. The system architecture is based around the use of Java Message Service (JMS) [37] which is a Message Oriented Middleware (MOM) API. Using JMS enables loosely coupled, reliable and asynchronous distributed communication between the application components in an event-based manner [42]. This helps to eliminate unnecessary polling activities and results in a system that quickly and efficiently is able to respond to new events. JMS provides messaging primitives for various patterns of communication including point-to-point (JMS queue) and publish-subscribe (JMS topic). Figure 3.3 shows how the JMS message bus acts as the central point of communication to integrate the different systems. Using a message bus enables the components to work together and allows components to easily be added and removed.

The system design is based on well known patterns used in the design of MOM based systems and other enterprise integration platforms [31]. A JMS message broker is needed to provide a platform for the deployment of JMS based solutions. Apache ActiveMQ [21] is an open source implementation of JMS that runs on the home server and provides the infrastructure of the messaging system. The ActiveMQ message broker handles storage and reliable delivery of all the messages through either of the JMS messaging primitives.

Figure 3.4 shows an overview of the mobile device software, the control server and the burglar alarm server. The mobile devices keeps track of the various services, locations and location providers available to them. They also provide various GUIs to allow user interaction with the rest of the MCS. The control server is able to keep track of multiple mobile devices using different services and potentially being located in different locations.

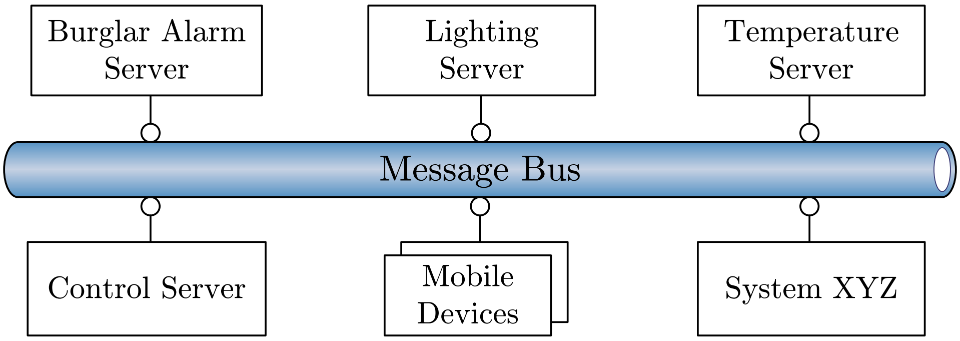


Figure 3.3: System integration using a message bus

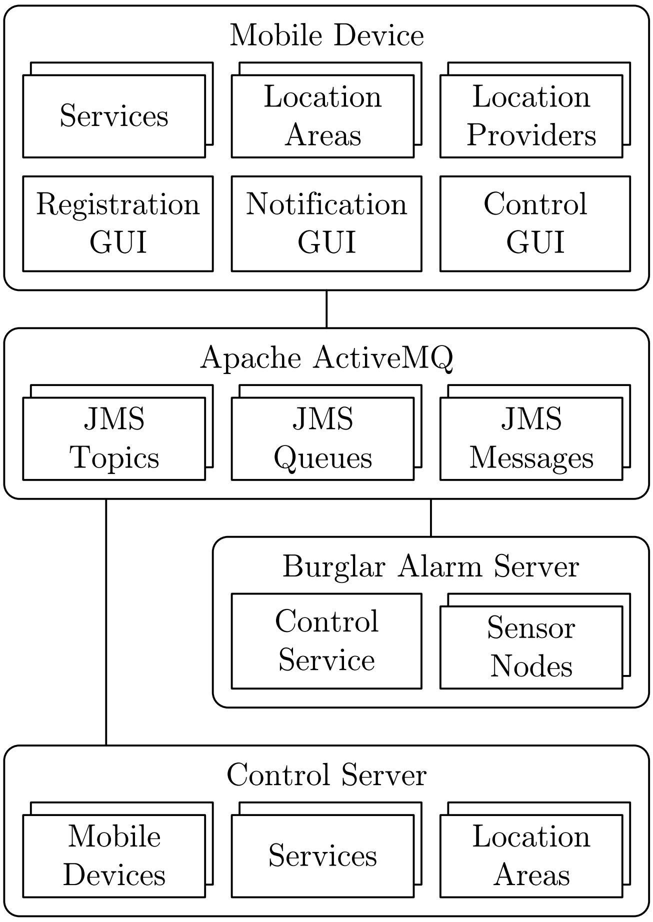


Figure 3.4: System architecture overview

A collection of multiple locations usually result in what is known as the home area. The alarm server simulates an interface to an actual burglar alarm system and the underlying alarm sensors. It has a control service which accepts commands to activate and deactivate the system, as well as querying the current status. The mobile devices, control server and the alarm server are tied together by JMS message queues and topics provided by Apache ActiveMQ.

All communication and notification events in the system eventually ends up passing through ActiveMQ. That might sound like a guaranteed way to induce scalability and reliability (single point of failure) problems. However, multiple JMS message brokers on different servers can be configured to provide fault tolerance, high availability and load balancing, if needed. Such a change would require configuration changes , but few, if any, code changes would be needed. Regardless, communicating using JMS on a local system (like the control server and alarm system) is by design very performance efficient. Given some decent server hardware there are few reasons to expect performance problems unless the number of services and associated mobile devices becomes very high.

3.3 Communication

The communication between the different components in the system makes use of the JMS message primitives provided by the ActiveMQ message broker. On the mobile device the Overlynx background service maintains a constant communication link with the JMS message broker located on the home server. The IP address or host name of the home server is assumed to be stored on the mobile device so that a connection can be established. Standard Java JMS client libraries refused to work out of the box on Android because they required an unavailable package. To solve the problem the Extensible Messaging and Presence Protocol (XMPP) [25, 12] supported by ActiveMQ was used as a replacement. Using the Smack API [44], which provides an implementation of the XMPP standard, allowed connections to multiple JMS topics and queues on the home server to be established.

To be able to establish a connection to the home server the mobile device requires some sort of network to relay its communications. GSM and UMTS network operators provide their users with data services so that mobile phones can access the Internet. The cost of such services are usually based on the amount of data transferred, but some network operators also offer unlimited data plans for a fixed price. The event-based nature of the MCS means that it will only transmit data when significant events take place. This should help reduce the costs incurred by mobile data services. Android will also be able to take advantage of WiFi network connections, if one is available, to further reduce costs.

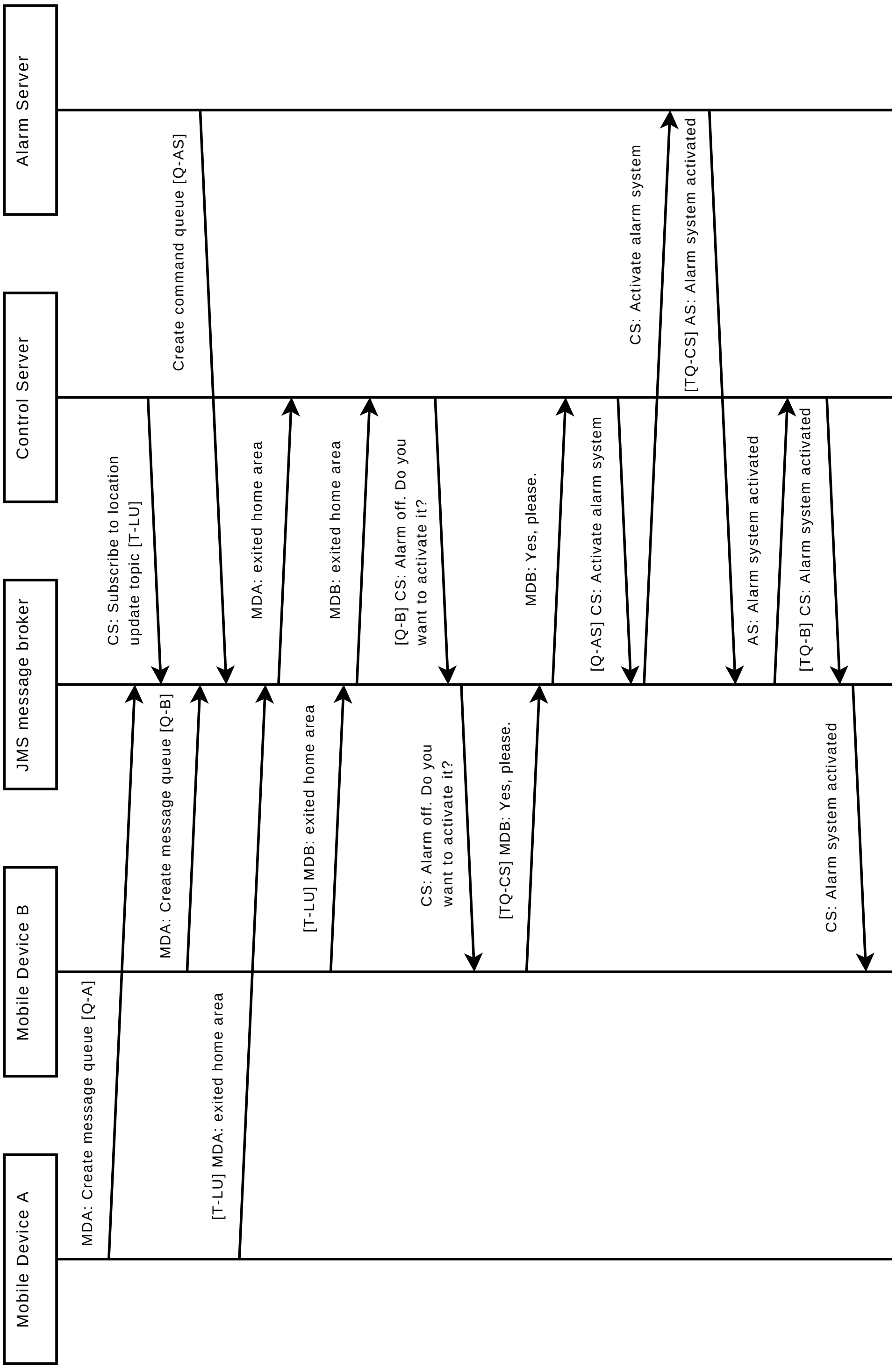

Mobile Device A
Mobile Device B
JMS message broker
Control Server
Alarm Server
MDA: Create message queue [Q-A]
CS: Subscribe to location update topic [T-LU]
MDA: Create message queue [Q-B]
Create command queue [Q-AS]
[T-LU] MDA: exited home area
MDA: exited home area
[T-LU] MDB: exited home area
MDB: exited home area
[Q-B] CS: Alarm off. Do you want to activate it?
CS: Alarm off. Do you want to activate it?
[TQ-CS] MDB: Yes, please.
MDB: Yes, please.
[Q-AS] CS: Activate alarm system
CS: Activate alarm system
[TQ-CS] AS: Alarm system activated
AS: Alarm system activated
[TQ-B] CS: Alarm system activated
CS: Alarm system activated

Figure 3.5: Example sequence of communication

Figure 3.5 shows an example of a sequence of communication between two mobile devices (MDA, MDB), a JMS message broker, control server (CS) and alarm server (AS). JMS queues and topics are enclosed in square brackets where the prefix T means a topic, Q is a queue and TQ is a temporary queue (used in request-reply sequences). Both mobile devices start by creating individual message queues that can be used to send messages back to the mobile device in question. When the location update service on the mobile device detects that it has left the home area it will send an "area exited" event notification to the JMS location update topic. The control server, subscribing to this topic, makes note of which mobile devices are inside and outside the home area at any given time. The notification from MDB, being the last to exit the home area, causes the control server to send a request back to MDB asking if he wants to activate the alarm system. MDB replies with a yes answer to a temporary queue (TQ-CS) the control server is listening to. The control server then issues an activation command to the alarm server which results in the activation of the alarm system.

3.4 Security considerations

Although security has not been the main concern during the design and implementation of the MCS, it is important to develop an understanding of the ramifications of this system. An important concern is that someone might steal your mobile device and use it to disable the alarm in your house. As a result of this the use of the auth(entification) policy for alarm deactivation has been recommended. It would also be a good idea to strictly enforce this by preventing users from switching to a less secure policy. Alternatively a warning and confirmation dialog could be displayed if someone tries to switch to a policy that might result in reduced security.

Another danger is that people may get so used to the alarm system being automatically activated that they take it for granted. One day a user will forget his mobile device at home and the alarm will not activate automatically. A solution to this problem might be to install more sensors at the entrances to monitor where users are without the need for mobile devices. This approach will pose an entirely new set of challenges.

It is practically impossible to always reliably identify users without embedding a chip under their skin. If motivated, a user can easily lie to the system about his location by leaving his mobile device behind, turning it off or handing it over to someone else. Hopefully the benefits the system brings to the users, will motivate them so that they do not deliberately try to subvert it.

The most common criticism against location-based services is related to privacy issues. The act of disclosing your own locations causes great concern for many users. In the MCS location information is disclosed to the home server and potentially other users of the system. The precision of the location information in this system is relatively coarse which reduces the potential for abuse. The current implementation of the MCS does not apply any encryption on the transmitted data, which is likely to cause problems in real world applications. It does include rudimentary authentication, but this mechanism does not ensure the safety of the location information nor any other information transmitted across the network.

A potential attack vector[1] on location-based services is location spoofing. Applications using WiFi location information can easily be tricked into believing they are somewhere other than their physical location. By impersonating one or more existing WiFi access points, a mobile device relying on this information will think that it is somewhere else entirely. The iPhone and other devices based on the Skyhook Wireless [10] WiFi database has been shown to be vulnerable to this technique [50]. One way to prevent this is to use location information from additional sources.

Using the iwlist Linux command to scan for APs revealed that it is possible to uniquely identify each AP by its MAC address. If the AP does not broadcast its ESSID it should still be possible to identify the home area AP by its MAC address. Other unique features, such as the frequency/channel that the AP operate on, what bit rates it supports and the security mechanisms it supports, could possibly also be used to establish that the correct AP has been found.

If an attacker somehow managed to replicate the AP by spoofing the MAC address, ESSID and the other unique features of the original home area AP other measures would be needed to ensure that the mobile device software does not confuse this duplicate AP as the actual home area AP. One solution is to require that the mobile device first successfully authenticates with the AP. Duplicating the authentication details of an AP is probably a much more difficult process.

3.5 The control server

The control server stores information about the available services, associated mobile devices and location areas. It also keeps track of which mobile devices are inside and outside of each area and dispatches event notifications depending on the location of the mobile devices, the state of the available services and the current policies for each service. The home area can in some cases consist of multiple locations. An example of a location is the area covered by a single WiFi access point. The combined area covered by multiple WiFi access points makes it possible to define a single home area for large buildings as well.

[1] An attack vector is a path or means by which a malicious person can gain access to a computer or network server in order to deliver a payload or malicious outcome.

3.5.1 Policies for activation and deactivation

A few approaches to how services should be activated and deactivated when a location change is detected, has been developed. The user can choose between several different policies concerning activation/deactivation of the alarm system. Upon entry into the home area, the mobile device could do one of the following:

1. Automatically deactivate the alarm system without user intervention,
2. request user to acknowledge deactivation, or
3. request user to acknowledge deactivation with a PIN code.

Policy for activation is more or less reverse, except that the system will have to consider multiple mobile devices associated with the home area:

1. Automatically activate the alarm system without user intervention when the last mobile leaves the home area,
2. request the last user that left, to acknowledge activation (used by the example in Figure 3.5), or
3. request the last user that left, to acknowledge activation with a PIN code.

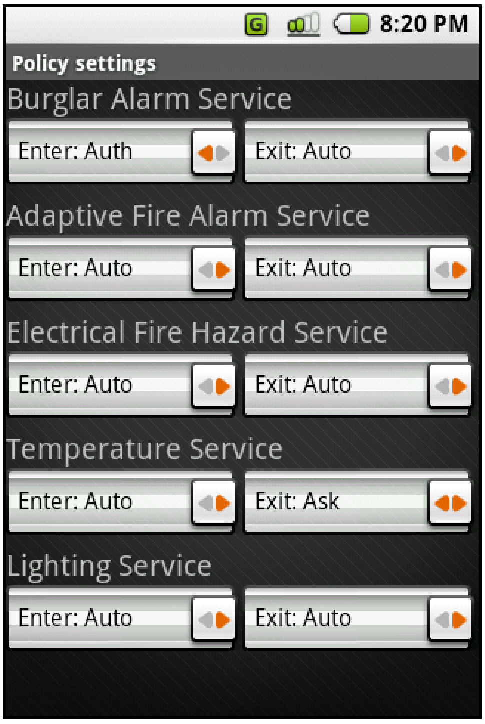
If the mobile device is stolen, deactivation policy 1 and 2 would allow a malicious person to disable the alarm system and enter the house without the need to know the PIN code. For this reason deactivation policy 3 should be the default choice for the burglar alarm service. Types of services that do not require a high level of security could take advantage of the two first policies to make the system less cumbersome to use. Examples of services that are suitable for either of the policies includes energy conservation services that automatically adjust the temperature in the house and turn off electrical appliances that the user has forgotten to power off, including lighting.

The advantage of these policies is that they give users flexibility and control over how the state of the service is changed. The above policies can easily be generalized so that they can be applied to other kinds of services. Figure 3.6 shows an example GUI in Android using the generalized set of policies presented below:

auto – change the state of the service without user intervention

ask – ask user to accept or reject the state change

auth – require that the user authenticate before any changes takes place



Policy settings

Burglar Alarm Service

Enter: Auth
Exit: Auto

Adaptive Fire Alarm Service

Enter: Auto
Exit: Auto

Electrical Fire Hazard Service

Enter: Auto
Exit: Auto

Temperature Service

Enter: Auto
Exit: Ask

Lighting Service

Enter: Auto
Exit: Auto

Figure 3.6: Example of a generalized policy configuration screen

3.6 The burglar alarm server

The burglar alarm server publishes event notifications to a JMS topic that zero or more subscribers can listen to. It publishes the alarm state change notifications like activated, deactivated and triggered as shown in Figure 3.7. To make it possible to activate and deactivate the alarm by a mobile device the service also listens to a JMS queue for control commands. This allows a mobile device to activate/deactivate and query the status of the alarm using the request-reply messaging pattern. The the burglar alarm server supports two modes of simulation:

Quiet causes the alarm server to only respond to activation/deactivation and status queries received through its JMS command queue.

Noisy will change the alarm state at random intervals to simulate that someone else is remotely controlling is. This is an extension of the quiet mode so it also accepts commands received by JMS.

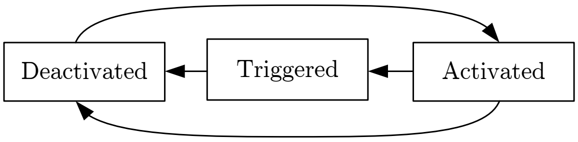


Figure 3.7: State machine of the simulated burglar alarm system

3.7 The Android mobile client

The Android mobile client, also known as the droid, is the software that runs on the mobile devices. Its goal is to provide the users with the interface they need to manage services and the logic needed to detect when users enter and exit the home area. This software has only been tested in an emulator as no real mobile phones running Android has been released to the market yet. The mobile client software is divided into two separate programs:

Overlynx is a background service that is automatically started when the mobile device is powered on. It maintains a connection to the remote JMS server and handles any messages and event notifications it receives. It also monitors the location of the mobile device and notifies the control server when the user changes location.

MCS Control is the application that allows users to directly interact with services on the home server. Policy changes and other settings are also managed through the user interface of this application.

Due to the beta state of the Android SDK some of the APIs, like those for Bluetooth and WiFi, have not yet been finalized. This problem has been partially solved by writing a custom WiFi API that emulates the behavior of a real WiFi adapter. The location provider API is available along with an emulated GPS location provider. The Cell-ID API is also available, but the information it provides is hard coded in the emulator.

For any number of reasons it might sometimes be impossible to acquire accurate information. Other times the mobile device might be unable to communicate with the home server or it might even be turned off completely. The system keeps track of how long ago it last communicated with each device. If it does not hear from a device in a while it will assume that the user has either turned it off on purpose or that it has run out of battery or that its broken. Regardless of the reason the system will change the device status to unknown and deregister it so that it does not interfere with the usage of the system. If it hears from the device again it will automatically register it after it has learned about its current location.

3.7.1 Avoiding the ping-pong effect

A location can easily be defined as a perfectly circular area, and by using GPS it is possible to find out if you are inside or outside this area. The system itself is based on discrete states such as inside and outside, but the real world is full of different colors and shades. If the user is standing exactly at the boundary of the home area, the location provider might determine that the user is inside the area one second and outside the next. If the location provider is able to provide location information in sufficiently small intervals the system might see the user bouncing in and out of the area faster than it can handle. To prevent this so called ping-pong effect a threshold value is used in order to create a buffer zone as shown in Figure 3.8. The location provider update interval will also have to be set to a sufficiently high value. If the connection to the WiFi access point (AP) is lost the user is determined to be outside the area. To be considered inside the area again the signal strength to the associated AP must increase until it is above a specified threshold value.

By using a circular area it is easy to adapt the concept to GPS as well. If the user is more than a certain distance d_{max} away from the center of the circle the system will consider the user to be outside the area. At this point the user will only be able to enter the area if he reduces his distance to the center until it is less than $\lambda \cdot d_{max}$ where $\lambda \in [0, 1]$. In the current implementation the threshold coefficient is fixed at $\lambda = 0.9$. This creates a buffer zone using 10% of the total radius. The system could potentially allow the user to manually adjust λ if he is not satisfied with the default value. The function in Algorithm 1 is called every time a location update notification is received from the GPS location provider. The current coordinate (x_0, y_0) provided by the GPS is passed as a parameter to the function. It will return true only if the location of the user has changed from inside the area to the outside, or the reverse.

3.7.2 GPS

The Android emulator allows a file with pre-recorded track points to be installed so that it can emulate a real GPS and make applications believe that it is actually moving. Using this technique, a track was recorded with a real GPS and uploaded to the emulator. To be able to find out if a user is inside or outside the home area one must first find a suitable method to define this area When a GPS is used the home area is defined as a circle with radius r and center coordinate (x_c, y_c). Following the recommendation of the previous section the implementation also makes use of a threshold value to avoid the ping-pong effect.

In order for the system to be of any use to a specific household it must be possible for users to specify a home area that suits their needs. Figure 3.9

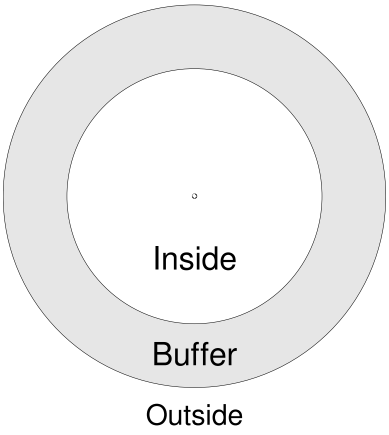
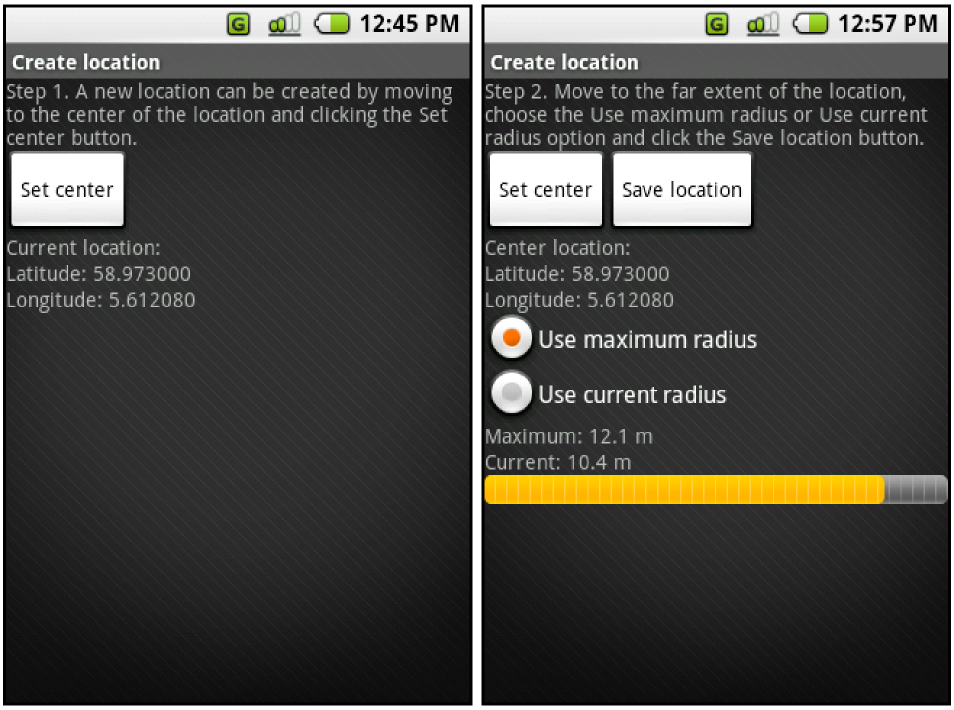


Figure 3.8: Circular location area with a buffer zone

Algorithm 1 Evaluate a GPS location update.

```
class GPSLocation
    $\lambda \leftarrow 0.9$
    $inside \leftarrow false$
     GPSLocation($latitude, longitude, r_{max}$)
        $x_c \leftarrow latitude$
        $y_c \leftarrow longitude$
        $d_{max} \leftarrow r_{max}$
    function gpsLocationUpdate($x_0, y_0$)
        $d \leftarrow$ distance between $(x_c, y_c)$ and $(x_0, y_0)$
        if $inside$ and $d > d_{max}$ then
            $inside \leftarrow$ false                    {User left area}
            return true              {Indicate that inside has changed}
        else if not $inside$ and $d < \lambda \cdot d_{max}$ then
            $inside \leftarrow$ true                  {User entered area}
            return true              {Indicate that inside has changed}
        return false                          {State remains unchanged}
```



Create location

Step 1. A new location can be created by moving to the center of the location and clicking the Set center button.

Current location:
Latitude: 58.973000
Longitude: 5.612080

Create location

Step 2. Move to the far extent of the location, choose the Use maximum radius or Use current radius option and click the Save location button.

Center location:
Latitude: 58.973000
Longitude: 5.612080

● Use maximum radius

○ Use current radius

Maximum: 12.1 m
Current: 10.4 m

(a) Before center is set

(b) After setting the center

Figure 3.9: Specifying a GPS location

shows the interface that is used to create a GPS location. The first step is to move to the location where you want the center of the circle to be and then click the "Set center" button shown in Figure 3.9a. This will assign the latitude and longitude of the current GPS position to (x_c, y_c). The application will then continually update the distance between the current location and (x_c, y_c) and show the result as in Figure 3.9b. If "Use maximum radius" is selected (default) when the "Save location" button is clicked, r will be set to the maximum distance measured from the center. Otherwise, the "Use current radius" option has been selected and r is set to the current distance.

A disadvantage of this approach is that you will most likely fail if you want to specify the center of the circular area in the middle of your house. Because regular GPS devices are unable to provide reliable location information inside a house your only alternative would be to climb up on the roof. An alternative approach would be to tell the user to walk around the perimeter of where they want the home area to be. The application would store all the coordinates along this trip and use a smallest enclosing circle algorithm [34] to fit a circle around the coordinates.

To actually collect the GPS location information the mobile client first has to register with Android's location provider API. The Overlynx background service starts by querying for an appropriate location provider. When one is found it tells Android that it wants to receive updates for this provider on a regular basis. Each time a location update is received it calls the gpsLocationUpdate() function presented in Algorithm 1 where $d_{max} = r$. The current location (x_0, y_0) is passed as argument to the function. When this method returns true it checks the value of the *inside* variable and reports the new location back to the home server.

3.7.3 WiFi

The location detection algorithm for WiFi areas uses the exponential moving average formula in Equation 3.1 to smooth out large variations in the reported signal strength.

$$d_t = \alpha \cdot s_t + (1 - \alpha) \cdot d_{t-1} \tag{3.1}$$

Where d_t is the average signal strength at time t, s_t is the reported signal strength and $\alpha \in [0, 1]$ is the weighting factor. Algorithm 2 shows the location update method used for WiFi areas. It contains a self-adapting algorithm that automatically determines the maximum and minimum signal strength values. The test results in Section 4.1 shows why a simple threshold based algorithm would not work and why a more advanced algorithm is needed. When it looses contact with the AP it assumes that the user has reached the edge of the coverage area. This algorithm will have some problems if there are blind spots in the coverage area of the wireless network inside the area the user intended to be part of the home area. Any place and any time the mobile device loses the connection to the AP it will assume that the user has left the home area. This will also falsely trigger a location exit event if the AP is physically turned off or otherwise disabled.

A solution to this problem is to use a timeout mechanism to delay the location exit event until one can be relatively certain that the user has indeed left the area. In most cases this will solve the problem where the user temporarily loses contact with the AP while he is still inside the area. A simple solution for permanent blind spots is to install another AP that is able to cover this area and consider the user inside if he has contact to either of the APs. Special sensors at each entrance to the home area could also be installed to detect when people enter or exit. It is recommended that an uninterruptible power supply (UPS) is installed to prevent short-lived power outages from shutting down both the AP and the home server.

3.7.4 Power saving

To support as many different mobile phones as possible the use of WiFi has been an important part of the implementation. Also Cell-ID information of all the cells that overlap with the WiFi coverage area could be registered by the system. If the mobile device is outside all of the registered cells it can safely disable WiFi to save power as shown by Algorithm 3. Whenever the device enters one of the registered cells it can automatically re-enable the WiFi. The same logic can also be used in conjunction with Bluetooth and GPS to preserve battery life. Using Cell-ID is practically free in terms of power consumption, as it is part of the existing cellular network protocols.

3.7.5 Location manager

To make use of the GPS and WiFi location classes and update functions we use an event-driven approach encouraged by the Android platform. In Algorithm 4 we begin by registering with Android's GPS location and WiFi information providers. By doing this we tell Android to get back to us when either the GPS location or WiFi information is updated.

The GPS event handler checks every registered location and calls the location update function to see if the user changed location. The WiFi event handler has an additional check to ensure that only the WiFi location that belongs to the AP identifier by p is checked. If the location of the user has changed it will dispatch an event notification to the home server, or more specifically the JMS location update topic.

At this level in the system there is no logic that says that the user entered or left the home area because the home area concept does not exist. As far as the mobile client is aware it only keeps track of different GPS and WiFi locations areas. The part that checks if a user is inside or outside the home area is located on the server. The server code has been made deliberately simple, so it only checks if the user is in at least one of the defined locations. If not then the user is outside the home area. Other more complicated scenarios are possible. For example one location could be assigned to a garage area which is not part of the home area. This location may overlap geographically with some part of the home area, but the location itself would in this case not be part of the set of locations that define the home area. A garage door opener service could then check if the user is in the garage area and open the door accordingly.

Algorithm 2 Evaluate a WiFi location update.

```
class WiFiLocation
    $d_t, s_{t-1}, s_{best}, s_{worst} \leftarrow \varnothing$
    $\lambda \leftarrow 0.9$
    $\alpha \leftarrow 0.3$
    $inside \leftarrow$ false
     WiFiLocation($networkid$)
        $ap \leftarrow networkid$
    function wifiLocationUpdate($s_t$)
        if $s_t = 0$ and $inside$ then
            if $s_{t-1} < s_{worst}$ then
                $s_{worst} \leftarrow s_{t-1}$                    {Reduce worst value}
            $d_t \leftarrow s_{worst}$                    {Reset average to the worst case}
            $inside \leftarrow$ false                    {User left area}
            return true                    {Indicate that inside has changed}
        else if $s_t \neq 0$ then
            if $d_t = \varnothing$ then
                $d_t, s_{best}, s_{worst} \leftarrow s_t$
            $d_t \leftarrow \alpha \cdot s_t + (1 - \alpha) \cdot d_t$                    {Smooth out variations}
            if $s_t > s_{best}$ then
                $s_{best} \leftarrow s_t$                    {Increase best value}
            $s_{t-1} \leftarrow s_t$                    {Update last value}
            if not $inside$ and $d_t > \lambda \cdot (s_{worst} - s_{best}) + s_{best}$ then
                $inside \leftarrow$ true                    {User entered area}
                return true                    {Indicate that inside has changed}
        return false                    {State has not changed}
```

Algorithm 3 Power saving with Cell-ID.

```
$H \leftarrow$ set of cells covering the home area
$inside \leftarrow false$
upon updated set of cells $C$ currently in range do
    if $inside$ and $H \cap C = \emptyset$ then
        Disable WiFi, GPS and Bluetooth if no other applications are
        using them.
        $inside \leftarrow false$
    else if not $inside$ and $H \cap C \neq \emptyset$ then
        Enable WiFi, GPS and/or Bluetooth if needed for localization.
        $inside \leftarrow true$
```

Algorithm 4 Mobile client location provider event handlers.

```
$L_{gps} \leftarrow$ set of all GPS locations
$L_{wifi} \leftarrow$ hash table of all WiFi locations
Initialization:
    Register with Android's GPS location provider
    Register with Android's WiFi connection information provider
upon GPS location updated to $(x_0, y_0)$ do
    foreach GPS location $a$ in $L_{gps}$ do
        if $a$.gpsLocationUpdate($x_0, y_0$) then
            if $a.inside$ then
                notify home server that user entered GPS location $a$
            else
                notify home server that user exited GPS location $a$
upon WiFi signal strength for AP $p$ updated to $s_0$ do
    $a \leftarrow L_{wifi}.get(p)$
    if $a \neq \varnothing$ and $a$.wifiLocationUpdate($s_0$) then
        if $a.inside$ then
            notify home server that user entered WiFi location $a$
        else
            notify home server that user exited WiFi location $a$
```

Chapter 4

Testing and simulation

To test the software developed in this project some real world data has been collected. This has been done using Kismet [7], an IEEE 802.11 wireless sniffer application that can collect signal strength information of WiFi access points (AP). Using the GPS and the WiFi network car,d Kismet was able to record packets sent from the local AP and tag them with the current geographical location. These data were pre-processed and the distance from the AP to each received packet was calculated. The following hardware has been used during these tests:

- GlobalSat Data Logger GPS DG-100 60 samples/minute
- Linksys WRT54GL Access Point
- Dell laptop with Intel PRO/Wireless 2200BG WiFi card

4.1 WiFi coverage area

Section 3.7.4 of the report discuss how the coverage area of WiFi can be used to define a location. In this part of the report we investigate the WiFi coverage area in more detail. Figures 4.2, 4.3, 4.4 and 4.5 shows how the signal strength vary depending on the distance to the AP. Higher signal level means better reception. As expected the graphs show that the signal level increases the closer to the AP you get. Overall the signal level appears to vary somewhere between -20 and -85 dBm.

The first three graphs are measurements taken from the same area. In Figure 4.2 the AP is located in the basement while in Figure 4.3 it is moved up one floor. To collect the signal data the same route was taken to ensure that any variations induced by environmental factors were similar in both measurements. The route went around the house a couple of times and up and down an adjacent street to get readings from multiple directions. From the graphs it appears that the wooden walls gives a relatively good coverage area outside the house. The difference in signal level is relatively significant where having the AP on the first floor also gives a much better range. The range between 59 and 72 meters on the first floor AP is blank because of a building that is blocking the path. Looking at the large variations in signal level even at the same distances confirms that the coverage area of WiFi networks are far from a uniform circle. Using the gpsmap tool provided by Kismet, the coverage area was estimated as shown in Figure 4.1. The bright dots and paths inside the coverage area are locations where packets were received.

In Figure 4.4 the AP is moved outside the house so that it has a direct line of sight at all times. The variations are much lower and it is not until about 120 meters that you are at risk of losing the connection. Being that the AP is in line of sight at all times one might have expected even less variations. However, various other WiFi networks and environmental conditions cause a significant drop in the signal level at about 20 meters.

The data for Figure 4.5 has been recorded in a rural area partly inside a forest. It exhibits a much more linear degradation in the signal level which could very well be attributed to the lack of other WiFi networks or any other interfering electrical equipment for that matter. The gap at the left of Figures 4.2 and Figure 4.5 is a result of not being able to sample data that close to the center. On each graph there is a sharp drop in the signal strength at around 15 meter. If one was to set a threshold value at some specific signal level it might be at -65 dBm, -55 dBm, -45 dBm and -40 dBm for Figures 4.2, 4.3, 4.4 and 4.5, respectively.

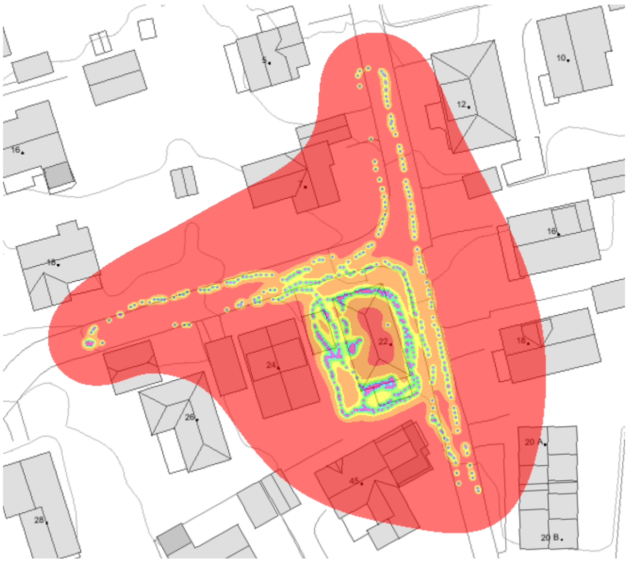


Figure 4.1: Estimated WiFi coverage area in suburban area with AP placed in basement

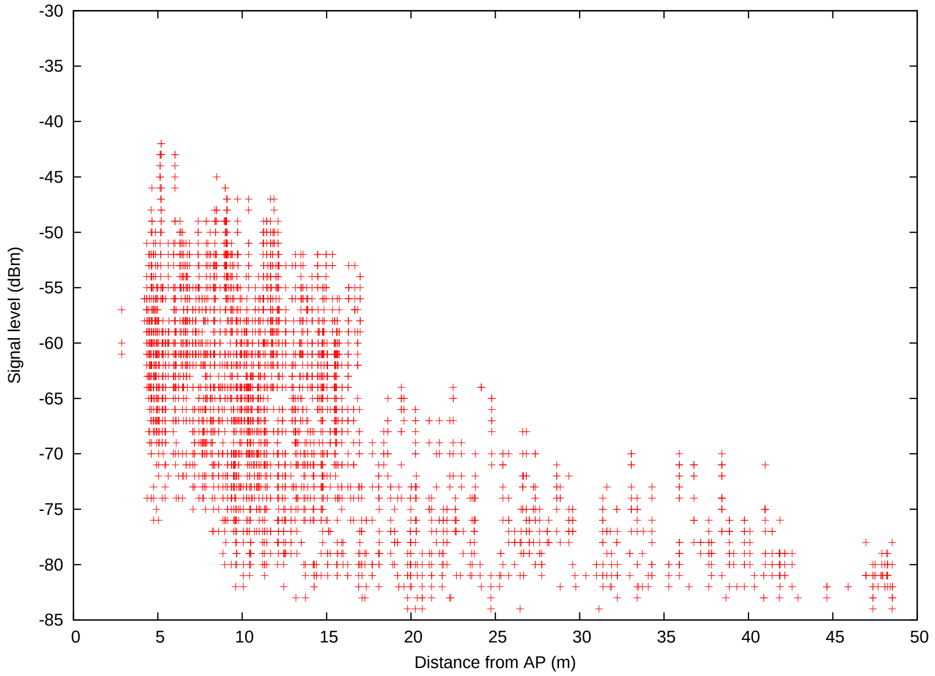


Figure 4.2: WiFi signal plot in suburban area with AP placed in basement

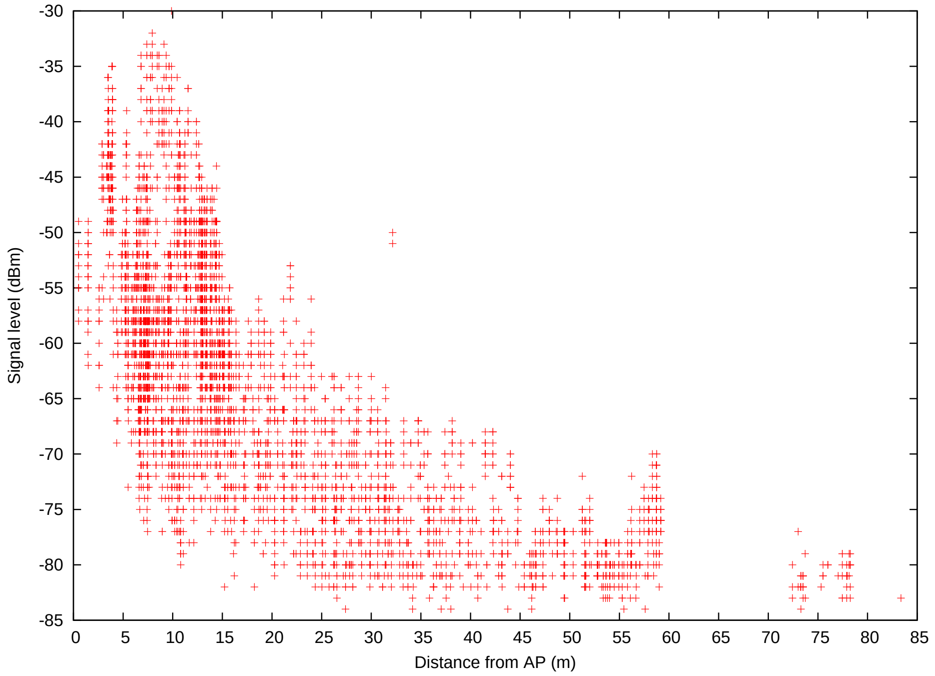


Figure 4.3: WiFi signal plot in suburban area with AP placed on first floor

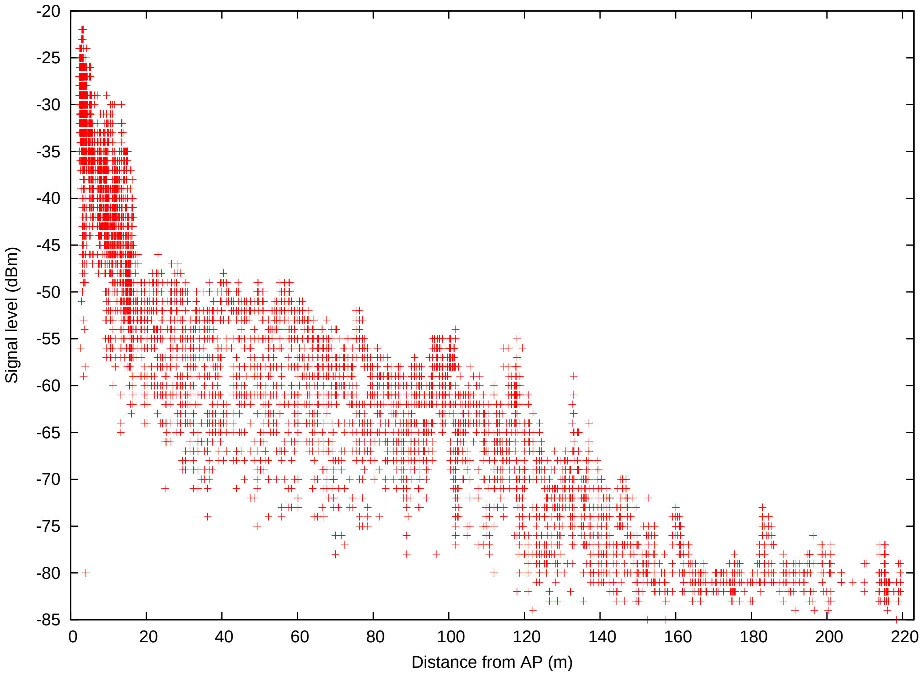


Figure 4.4: WiFi signal plot in suburban area with AP in line of sight

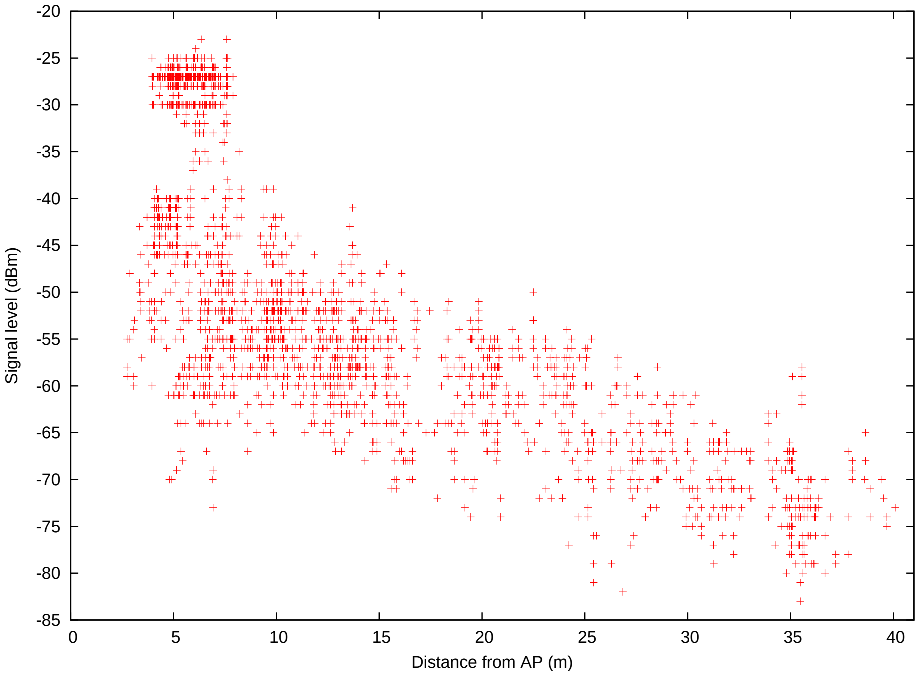


Figure 4.5: WiFi signal plot in rural area

The measurement results above showed that setting a simple threshold value can be a difficult and error prone task. Because the signal level varies so much and because it most often does not degrade linearly another approach must be taken. As described in Algorithm 2 an self-adaptive approach was chosen. To test this algorithm in practice the WiFi simulator server was used to collect signal strength values during a trip. The WiFi location update algorithm was then applied to this collection and the result is shown is Figure 4.6 The thin line is the signal strength and you can see the vertical lines where it loses connection with the AP and the signal level is reset to 0. The thick lines along the top and bottom represents the inside/outside status of the algorithm. It starts inside the home area and all the way up until it first loses connection to the AP. After that it takes a more careful approach and is stuck outside until the signal strength increases to a sufficiently high level.

The results look promising and the algorithm appeared to handle the multiple AP disconnections and varying signal strength without any major problems. This shows that the algorithm is able to adapt to situations where many fixes threshold values would easily break.

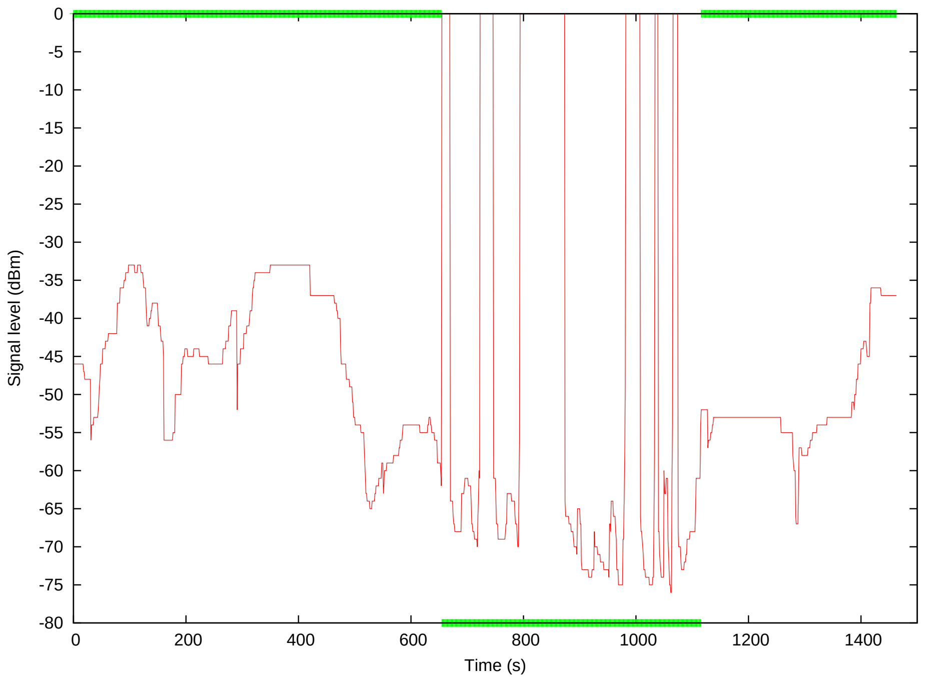


Figure 4.6: WiFi location update algorithm test

Chapter 5

Conclusion

This report has proposed a system that allows users to easily activate and deactivate a burglar alarm system using their mobile phone or other mobile device. Based on the location of the mobile devices, users will automatically receive notifications in the relevant situations allowing them to easily activate or deactivate the alarm system. The report has also looked at the possibility to integrate additional services into the system using a set of policies to determine how and when to activate and deactivate them. A number of challenges is involved in the design and implementation of a scalable and extensible distributed system as shown by this report. This includes algorithms that form a basis for location areas and algorithms that can help to reduce the power consumption in mobile devices.

5.1 Future work

Although great effort has been made to solve as many of the issues related to this system as possible, some remain unsolved. Most prominently is the issue related to unreachable devices. If a mobile device is turned off, runs out of battery or otherwise unable to establish a connection to the home server it cannot relay its position information. As far as the system is concerned the device and its user will remain at its current location until a certain period has passed. When the system has not heard from a device in a while it will remove it from the system so that it does not interfere with its regular operation. Even so this can result in some awkward situations. Consider a scenario where a mobile device of a user has run out of battery while he was outside the house. Soon after, this user enters his home together with his wife, but her mobile device is working fine. If his wife leaves and the alarm service is using the auto policy, the alarm will automatically be activated while he is still inside the house.

More work in securing the transmitted data is also needed. Currently all messages are transmitted across the Internet without any form of encryption.

To address the privacy concerns many users might have about sharing their location it is important to secure the data. This will also protect the MCS from several other types of attacks.

Another important step forward would be to get the mobile client software running on an actual mobile device. Mobile handsets running Android are scheduled for release in Q4 of 2008. Many indicators points to HTC as the first mobile handset manufacturer that will release a handset running Android. Version 1.0 of the Android SDK will also be released at the same time so parts of the current implementation of MCS is expected to break, as many APIs might change.

Bibliography

[1] BlueProximity. URL http://blueproximity.sourceforge.net/.

[2] CellDB. URL http://celldb.org/.

[3] CellSpotting, A Global Location Based Information Service. URL http://www.cellspotting.com/webpages/cellspotting.html.

[4] CellTrack. URL http://www.afischer-online.de/sos/celltrack/.

[5] floAt's Mobile Agent. URL http://fma.sourceforge.net/.

[6] GSMLoc: GSM cells, mapping CellID to GPS coordinates. URL http://gsmloc.org/.

[7] Kismet. URL http://www.kismetwireless.net/.

[8] metaquark Home Zone. URL http://metaquark.de/homezone/.

[9] Navizon - Virtual GPS for mobile devices and laptop computers. URL http://www.navizon.com/.

[10] Skyhook Wireless. URL http://skyhookwireless.com/.

[11] WiGLE - Wireless Geographic Logging Engine - Plotting WiFi on Maps. URL http://www.wigle.net/.

[12] XMPP Standards Foundation. URL http://www.xmpp.org/.

[13] 8Motions. OpenCellID. URL http://www.opencellid.org/.

[14] Apple. iPhone Dev Center. URL http://developer.apple.com/iphone/.

[15] Apple. Registered iPhone Developer Agreement. URL http://developer.apple.com/iphone/terms/registered_iphone_developer.pdf.

[16] Blaze. Mophun. URL http://www.mophun.com/.

[17] Jakub Marek Borkowski. *Applicable and accurate positioning techniques enable location-based services in cellular networks.* Ph.D. thesis, Tampere University of Technology, Finland, April 2008. URL http://www.cs.tut.fi/tlt/RNG/publications/docs/location/PhDThesis_JBorkowski.pdf.

[18] ACCESS Corporation. ACCESS Linux Platform (ALP). URL http://alp.access-company.com/.

[19] Intel Corporation. Place Lab. URL http://www.placelab.org/.

[20] Christine Dorffi. Comparing Mobile Platforms: Java ME and Adobe Flash Lite : Mobility Tech Tips. URL http://blogs.sun.com/mobility_techtips/entry/comparing_mobile_platforms_java_me.

[21] Apache Software Foundation. Apache ActiveMQ. URL http://activemq.apache.org/.

[22] Apache Software Foundation. Apache Harmony - Open Source Java SE. URL http://harmony.apache.org/.

[23] Apache Software Foundation. Maven. URL http://maven.apache.org/.

[24] Apache Software Foundation. Apache License, Version 2.0, 2004. URL http://www.apache.org/licenses/LICENSE-2.0.

[25] Jabber Software Foundation. RFC 3920 - Extensible Messaging and Presence Protocol (XMPP): Core, October 2004. URL http://www.ietf.org/rfc/rfc3920.txt.

[26] Joshua Gay. iPhone restricts users, GPLv3 frees them - Free Software Foundation, June 2007. URL http://www.fsf.org/iphone-gplv3.

[27] Google. Android. URL http://code.google.com/android/.

[28] Google. Google Maps with My Location (Beta). URL http://www.google.com/gmm/mylocation.html.

[29] IEEE 802.11 Working Group. IEEE 802.11-2007: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, June 2007. URL http://standards.ieee.org/getieee802/download/802.11-2007.pdf.

[30] IEEE 802.15 Working Group. IEEE 802.15.1-2005: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs), June 2005. URL http://standards.ieee.org/getieee802/download/802.15.1-2005.pdf.

[31] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional, October 2003. ISBN 0321200683.

[32] Harald Kunczier and Hermann Anegg. Enhanced cell ID based terminal location for urban area location based applications. In *Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE*, pages 595–599. 2004.

[33] Elena Malykhina. InformationWeek: GPS-Enabled Mobile Phones To Quadruple By 2011, November 2007. URL `http://www.informationweek.com/news/mobility/showArticle.jhtml?articleID=202801218`.

[34] Nimrod Megiddo. Linear-Time Algorithms for Linear Programming in R^3 and Related Problems. *SIAM Journal on Computing*, 12:759–776, November 1983. URL `http://link.aip.org/link/?SMJ/12/759/1`.

[35] Elin Melby and Tore Arthur Worren. Location Based Services in Cellular Networks. *ITS Norway Directory & Handbook 2005-2006*, pages 36–38, October 2005.

[36] Sun Microsystems. The Java ME Platform. URL `http://java.sun.com/javame/`.

[37] Sun Microsystems. Java Message Service (JMS). URL `http://java.sun.com/products/jms/`.

[38] Sun Microsystems. JavaFX Mobile. URL `http://www.sun.com/software/javafx/mobile/`.

[39] Sun Microsystems. JSR 914: Java(TM) Message Service (JMS) API. URL `http://www.jcp.org/en/jsr/detail?id=914`.

[40] Richard Monson-Haefel. Why Microsoft Loves Google's Android. URL `http://java.sys-con.com/read/467328.htm`.

[41] Motorola. MOTOMAGX. URL `http://www.motorola.com/content.jsp?globalObjectId=8411`.

[42] Gero Muhl, Ludger Fiege, and Peter Pietzuch. *Distributed Event-Based Systems*. Springer, Berlin, 1st edition, July 2006. ISBN 3540326510.

[43] Terje Moi Nilsen. Sales and Product Manager. NorAlarm. Personal communication.

[44] Ignite Realtime. Smack API. URL `http://www.igniterealtime.org/projects/smack/`.

[45] James M. Hasik Michael Russell Rip. *The Precision Revolution: GPS and the Future of Aerial Warfare*. Naval Institute Press, 2002. ISBN 1557509735.

[46] C. Sammarco, F. H. P. Fitzek, G. P. Perrucci, A. Iera, and A. Mollinaro. Localization Information Retrieval Exploiting Cooperation Among Mobile Devices. In *Communications Workshops, 2008. ICC Workshops '08. IEEE International Conference on*, pages 149–153. 2008. URL `http://cooploc.es.aau.dk/download/cooploc.pdf`.

[47] Stefan Steiniger, Moritz Neun, and Alistair Edwardes. Lecture Notes on LBS, Lesson 1: Foundations of Location Based Services, 2006. URL `http://www.geo.unizh.ch/publications/cartouche/lbs_lecturenotes_steinigeretal2006.pdf`.

[48] Symbian. Symbian OS. URL `http://www.symbian.com/developer/`.

[49] Adobe Systems. Adobe Flash Lite. URL `http://www.adobe.com/products/flashlite/`.

[50] Nils Ole Tippenhauer, Kasper Bonne Rasmussen, Christina Ppper, and Srdjan Capkun. Location Spoofing Attacks on the iPhone and iPod, April 2008. URL `http://www.syssec.ch/press/location-spoofing-attacks-on-the-iphone-and-ipod`.

[51] Emiliano Trevisani and Andrea Vitaletti. Cell-ID location technique, limits and benefits: an experimental study. In *Mobile Computing Systems and Applications, 2004. WMCSA 2004. Sixth IEEE Workshop on*, pages 51–60. 2004. ISBN 1550-6193.

[52] Trolltech. Qtopia. URL `http://trolltech.com/products/qtopia`.

[53] Alex Varshavsky, Eyal de Lara, Jeffrey Hightower, Anthony LaMarca, and Veljo Otsason. GSM Indoor Localization. *Pervasive and Mobile Computing*, 3:698–720, December 2007. ISSN 1574-1192.

[54] Andrew Y. Wang and Charles G. Sodini. On the Energy Efficiency of Wireless Transceivers. In *Communications, 2006. ICC '06. IEEE International Conference on*, volume 8, pages 3783–3788. 2006.

[55] Torbjorn Wigren. Adaptive Enhanced Cell-ID Fingerprinting Localization by Clustering of Precise Position Measurements. *Vehicular Technology, IEEE Transactions on*, 56:3199–3209, 2007. ISSN 0018-9545.

[56] Nathan Willis. Linux.com :: The iPhone SDK and free software: not a match, April 2008. URL `http://www.linux.com/feature/131752`.

[57] Yahoo! Yahoo! Developer Network Home. URL `http://developer.yahoo.com/`.