Informatica — Universiteit van Amsterdam

Bachelor Informatica
Universiteit van Amsterdam

Sadako - Securing a building using IEEE 802.11

Sybren A. Stüvel

January 29, 2012

Supervisor(s): Leen Torenvliet (UvA), Andy Pimentel (UvA)

Signed:

Abstract

This paper describes the Sadako project. It is a system for securing a building by using IEEE 802.11 network technology to determine the position legal occupants of said building. The alarm will be turned off in zones where those legal occupants are located, and turned on in other areas.

The paper is accompanied by the source code of the Sadako prototype, in the form of a Sadako-specific fork of Place Lab, the Sadako client, and the Sadako Alarm Management Server.

Contents

List of Figures

CHAPTER 1

Introduction

Nowadays, many different alarm systems are available. Their purpose is always the same: to allow authorized people more or less easy access to a building, while keeping out malefactors. When the building has been illegally penetrated, a warning signal should be sent to make this clear. Currently in use alarm systems identify authorized people by using keys, passwords, chip cards or codes, and accompanying protocols.

This document describes the Sadako project in this chapter. Chapter 2 describes the theory behind Sadako. The work we have done to create the initial software is described in Chapter 3, along with the experiments we performed and their results. Chapter 4 is dedicated to security and privacy issues. Future improvements are discussed in Chapter 5, followed by the conclusion in Chapter 6.

The software that is published at the end of the project will be a prototype of the Sadako system. It is not a publicly usable, user-friendly, fool-proof piece of software, but it is functional in all the areas described in this document.

We are well aware of the fact that men and women are generally equal. However, to keep notation short and clear, we use the female form when talking about a Sadako user.

1.1 Description of the project

Sadako can manage the alarm inside a building. WiFi access points are placed in tactical points across the building, and broadcast their identity through the ether. People walk through the building with Sadako-equipped PDAs. Such a PDA has a database of the positions and identities of the access points. By measuring the signal strengths it receives from those access points, the PDA can determine its position. This position is transmitted securely to the Alarm Management Server, which in turn ensures the carrier of the PDA can freely walk around the building without setting off the alarm, while ensuring the alarm is turned on in unvisited zones.

1.1.1 Motivation

Many alarm systems work without partitioning. This means that the identification of a single authorized person will turn off the alarm in the entire building. Other systems use partitioning with barriers between the partitions. Moving from one partition to the other requires the entire authentication protocol to be replayed. Using conventional systems, this is not only expensive in terms of required hardware, it is also the cause of neglect on the part of the users, leading to more unprotected areas than strictly needed.

With Sadako, this is all in the past. Partitions can be defined without a barrier between them, and people can walk around freely. The Sadako system will track her users, and switch the alarm based on their known position.

One of the future goals of Sadako is to make it run on as many different PDAs as possible. This will allow the users to Sadako-enhance their own PDA, lowering the acceptance threshold. For this reason, we must only use common hardware in this project. More precise results can be obtained when using specialized radio detection hardware, but that would defeat the point of reusing the hardware users might already have.

1.1.2 About the name

In the Japanese movie Ringu, one of the main characters is a girl called Sadako Yamamura, at some point described as "She hears everything. She never sleeps." Those properties are of course very useful for an alarm management system.

1.1.3 Definitions and abbreviations

The following definitions and abbreviations are used in this document.

AES Advanced Encryption Standard.

AMS Alarm Management Server.

CRL Certificate Revocation List, which contains the serial numbers of revoked SSL certificates.

GUI Graphical User Interface.

JVM Java Virtual Machine, the software required to run Java programs.

MITM attack Man In The Middle attack, where someone sits between the intended sender and recipient in an information stream, with the intend to sniff the data and retransmit it, possibly altered, to the intended recipient.

PDA Portable Digital Assistant. Used to be a small computer that could be used as an agenda and address book. Nowadays it is a common name for a hand-held computer.

RPC Remote Procedure Call.

RSS Received Signal Strength.

SSL Secure Socket Layer.

WiFi Short for Wireless Fidelity. Common name for the IEEE 802.11 wireless network technology.

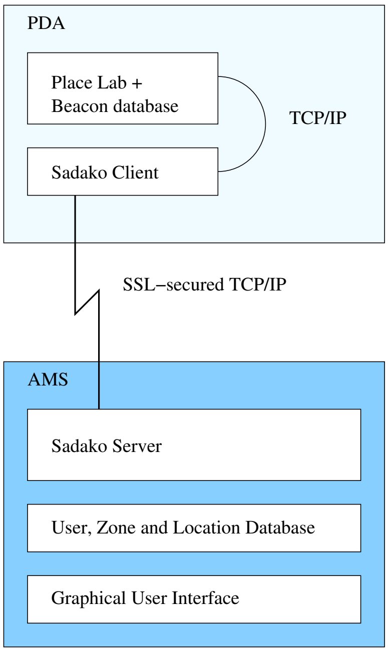

Figure 1.1: Architecture overview of Sadako

1.2 Outline of the Sadako Architecture

Sadako consists of many different components. An overview can be seen in Figure 1.1. In the following sections, we'll look at each component more thoroughly.

1.2.1 Components of Place Lab

Place Lab[7] is at the heart of Sadako. It allows the PDA to determine its position in a passive manner. This passiveness is important for the user's privacy - more on that in Chapter 4.

There are five main components[14] in Place Lab. The *spotter* connects to the WiFi card. It can post a request for a new scan of the radio-environment, and redirect the results of the scan to the next component. The *tracker* takes those measurements, looks up the beacon information in the *database*, performs calculations on their respective locations and signal strength, and ends up with a 3D coordinate. How this is done, is described in more detail in Chapter 2.

To be able to quickly look up information in the database, the tracker uses the *mapper*. Sadako's mapper is very simple, and the code could as well have been included in the database interface directly, but that would in our opinion alienate the Sadako-specific code from Place Lab.

The 3D coordinate obtained by the tracker is then sent to the *estimator*. This can then perform more calculations based on the current coordinate and remembered coordinates from the past. In its current form, the Sadako estimator also contains a simple HTTP server which allows the Sadako Client to request measurements and return the 3D coordinate.

To be system-independent, Place Lab is mostly written in Java [20]. We chose JamVM [3] as the JVM because it is small, provides everything required to run Place Lab, and most importantly, runs on our hardware. The only part that is native is coded in C, and functions as the interface between the 802.11 card and the Java code. It triggers a new scan of the surroundings, and returns the list of access points seen, together with their respective RSS.

1.2.2 Components of Sadako

Sadako is split up into two components. The positioning framework runs on the PDA. There are as many of those as there are legitimate people in the protected building. In general, there is only one Alarm Management Server (AMS) for each building.

PDA

Each legitimate person who wants to use Sadako has to carry a WiFi-enabled PDA. On this PDA are two programs that together make up the client-side of Sadako.

During the project, we used a Nokia 770 [5] as the PDA. It contains a 250 MHz ARM processor, a built-in 802.11g WiFi card, and it runs Linux. Nokia opened the specifications of the 770 for Open Source projects, and initiated Maemo [4] to serve as a development platform. This makes the 770 a pleasant device to work with, as all information is readily available, and a lot of software has already been ported. More information on the alterations performed on the PDA during the development of Sadako can be found in Section 3.1.1.

First the Sadako-specific build of Place Lab is started. The inner workings of Place Lab have already been explained in Section 1.2.1. After Place Lab, the Sadako client software is started. This software is written in Python[16] for optimal flexibility, speed of development, and platform independence.

Upon startup, the Sadako Client first connects to the AMS using TCP/IP. For this prototype, it is assumed that a method of communication has already been set up. In our prototype development environment, setting up this connection means connecting the Nokia 770 to the AP that is connected to the development machine. This can be done with the interface supplied by Nokia. Then Sadako secures the TCP/IP connection by a SSL handshake. If anything is not as it should be, the connection is immediately severed. The security aspects are further described in Section 2.3 and Chapter 4.

After a successful authentication of both the client and the server, the Sadako Client continues to poll the Place Lab instance running on the same PDA. The obtained coordinates are sent to the AMS using SadakoRPC. This continues until the Sadako Client is shut down.

Alarm Management Server

The AMS is the core of the Sadako infrastructure. It contains information about the alarm zones of the building, communicates with the various PDAs in the building, manages the alarms and provides a visual interface to the alarm system.

The software used in the AMS can all be found on a regular Linux system. For the development, we used Ubuntu Linux [10] version 5.10. It is an easy to use, user-friendly distribution that is also developer-friendly. We used PostgreSQL [8] 8.0 as the database server and wxWidgets for the graphical user interface (GUI). This ensures that the Sadako server not only looks good, but also that it is portable across many platforms.

The Sadako Server component is written in Python. It contains a Sadako-RPC server described in Section 3.1.2 used to communicate with the clients. The functions exposed through RPC allow the Sadako Clients to communicate their positions, and query the list of zones they are in. Whenever the position of a client is changed, both the database and the GUI are updated to reflect this. If a zone is turned on or off, the special hook functions `sadako.zone_occupied()` and `sadako.zone_emptied()` are called which can then handle anything required - switch alarms, lights, cameras, etc.

CHAPTER 2

Theoretical background

Of course, no Bachelor project is complete without a proper theoretical background. The basis for the Place Lab positioning is the signal attenuation of the IEEE 802.11 [17] radio signal. The other part of the positioning is the triangulation required to convert signal strengths to a position.

2.1 Radio signal attenuation

A radio antenna emits energy. This energy moves away from the antenna in three spatial dimensions, hence at a distance and using another antenna, less energy will be measured. Of course, this attenuation is influenced by a lot of factors - air pressure, walls, metal cupboards, etc.

The major source of interference [21] in the 2.4 GHz band comes from microwave ovens and alarm units used in buildings. Fortunately, during the development of the Sadako system, the alarms were turned off and there were no microwaves in the vicinity.

Getting the signal strength is not enough. In order to do something useful with it, it needs to be related to the distance by some function. The signal strength measured by the WiFi driver [2] is returned in dBm, a logarithmic scale relative to 1 mW. The strength s in dBm can be converted to the power P in mW as follows:

$$P = (1\,mW)10^{s/10}$$

To get proper results, and convert from dBm directly to a distance in meters, we measured using a more or less direct line of sight to a single access point. The average signal strength out of 30 measurements was taken every 1.2 meters[1], and plotted against the distance to the access point. The resulting measurements were fitted to a function using Mathematica. The fitted function and the measurements can be found in Figure 2.1 in blue. The exact measurements are in Table 2.1 The resulting function is approximately:

$$\begin{aligned} d'(s) &= -25155 - 1324s + 0.02s^3 + 10^{-4}s^4 - 29186\sqrt{-s} + 42804\log(-s) \\ d(s) &= max\,(0, d'(s)) \end{aligned}$$

[1] We used the ceiling tiles as a coordinate grid, and the tiles are 0.6 meter across

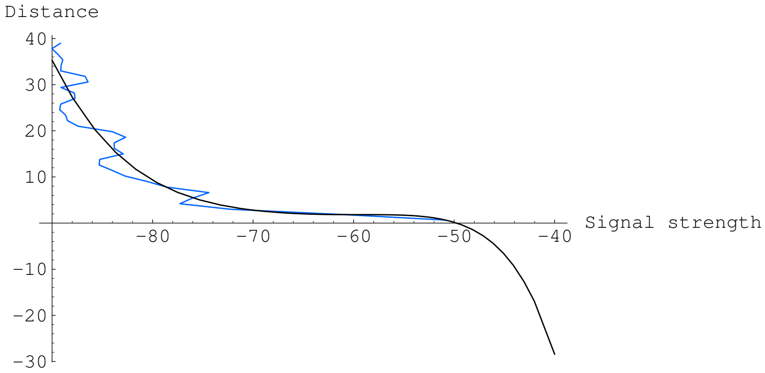

Figure 2.1: Measured signal strength in dBm against distance in meters. Every point is the average of 30 measurements.

which is not a nice function, but it fits the measured data and it also extrapolates the measurements well, at least for values not too far from the measured values. The function has reasonable values for a RSS of -100 to -20 dBm; values outside those limits will not be measured, because the signal then either becomes too weak to be scanned by the WiFi card, or would become stronger than transmitted by the AP[2]. The constants are rounded for printing purposes - the exact values can be found in the source code repository of `SadakoTracker.java`.

Unfortunately, the signal strength was too dependent on the environment. On the 5th of may 2006 we performed another measurement, and obtained the data plotted in green in Figure 3.6. It is clear that the distance function is unusable in a real situation.

2.2 Triangulation using weighed average

There are many ways to do triangulation. The most common ways use angles combined with distances [22]. Since the WiFi cards only report signal strength, we are unable to use those methods. The initial triangulation of Sadako used only the distance and not angles, and used a weight function based on this distance to calculate a weighed average of the coordinates of the access points.

This weighed average has a drawback and an advantage. The drawback is that the calculated position will always be within the convex hull of the access points. This is illustrated by the purple triangle in Figure 2.2.

Classical triangulation uses the distance to three known points to determine the position. This has the drawback that you need to have three access points in sight. The advantage of using a weighed average, is that it functions even if there is only a single access point to be seen. Of course, in such a case, the

[2] Measured using the AP at full power, and the Nokia 770 at a closer range than used when using the Sadako system

Dist	**27-04**	**05-05**
0 .0	-58.000	-57.133
1 .2	-51.667	-52.767
2 .4	-69.600	-58.767
3 .6	-74.967	-54.933
4 .8	-79.600	-59.767
6 .0	-72.500	-63.167
7 .2	-76.300	-67.800
8 .4	-80.900	-67.700
9 .6	-80.133	-67.267
10.8	-85.367	-65.367
12.0	-82.633	-66.567
13.2	-88.000	-66.600
14.4	-82.533	-66.867
15.6	-83.300	-57.800
16.8	-84.333	-58.967
18.0	-83.300	-63.967
19.2	-82.067	-65.367
20.4	-85.900	-68.533
21.6	-88.933	-72.567
22.8	-88.000	-71.433
24.0	-89.300	-68.233
25.2	-89.200	-69.233
26.4	-89.067	-68.933
27.6	-86.367	-66.867
28.8	-89.200	-66.567
30.0	-89.033	-71.967
31.2	-83.800	-66.800
32.4	-89.633	-63.433
33.6	-88.667	-65.600
34.8	-89.533	-66.333
36.0	-88.333	-74.133
37.2	-90.533	-68.067
38.4	-89.500	.
39.6	-88.800	.

Table 2.1: Measurements taken at different distances and different dates.

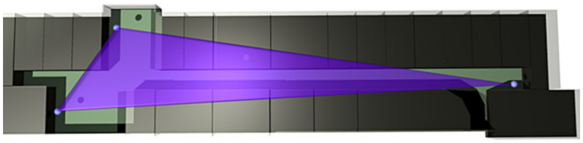

Figure 2.2: The area in which the coordinates will be calculated using weighed average. The blue dots are access points in use by Sadako.

calculated location is the location of that access point. This, however, does not necessarily have to be a problem for the Sadako project. After all, we are not directly interested in the position of the PDA, only in the zone the PDA is in. If the single visible access point is always in the same zone as the PDA seeing the access point, Sadako works as it should.

The initial weight function w used in the weighed average is

$$W(d) \quad = \quad \frac{4}{\max(d, 0.0001)}$$

Given a list of coordinates of visible access points $C = \{C_1, \ldots, C_n\}$ and their respective RSS $s = \{s_1, \ldots, s_n\}$, the coordinate of the PDA C_{pda} is determined using the function

$$C_{pda} \quad = \quad \frac{\sum_{i=1}^{n} W(d(s_i))C_i}{\sum_{i=1}^{n} W(d(s_i))}$$

2.3 Communication

A proper implementation of a communication protocol is crucial for the privacy of the users, and the security of the system. There are a couple of requirements for this communication:

Privacy Only the AMS may read the location information sent by the clients.

Server authentication The client must be sure of the identity of the recipient before sending any positional information.

Client authentication The server must be sure of the identity of the client before accepting any positional information.

MITM attacks A MITM attack should be detectable.

All those requirements can be taken care of by using SSL [13]. Sadako uses the popular OpenSSL [6] implementation to secure the communication between the PDA and the AMS. This at least ensures the *Privacy* and *MITM attacks* requirements are fulfilled.

The SSL protocol uses a sophisticated system of certificates. A certificate contains information about the owner, one or more signatures, and a serial number. Signatures are combined with a hash on the certificate data, hence a change in the data invalidates the signature. Those signatures are of paramount importance to Sadako, since they allow the AMS to authenticate the PDA without actual knowledge of any certificate of the PDAs. After the signature of the client certificate is validated, the serial number is checked against a CRL. If the certificate has not been revoked, the certificate's SHA1 fingerprint is looked up in the Sadako database. If it can be found and nobody else has connected with this fingerprint yet, the client is granted permission to continue. An overview of this can be seen in Figure 2.3.

At the same time, the PDA checks the AMS certificate's SHA1 fingerprint against a stored value. If the fingerprint matches this stored value, the PDA accepts the AMS's identity and will continue to communicate. It then proceeds

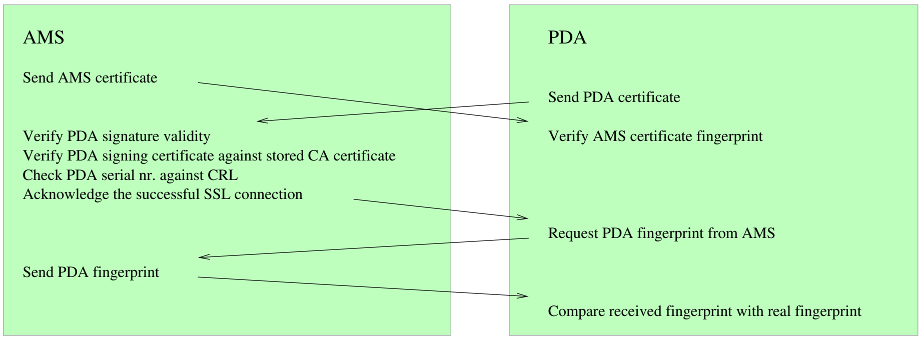
AMS
Send AMS certificate
Verify PDA signature validity
Verify PDA signing certificate against stored CA certificate
Check PDA serial nr. against CRL
Acknowledge the successful SSL connection
Send PDA fingerprint
PDA
Send PDA certificate
Verify AMS certificate fingerprint
Request PDA fingerprint from AMS
Compare received fingerprint with real fingerprint

Figure 2.3: Creating the SSL connection

by asking the AMS for its own fingerprint, which is then compared to the known value of the fingerprint. This is just a simple check to verify that the AMS really knows the identity of the PDA, and that the SadakoRPC system is functioning.

If at any point in time something is not as it should be, the connection is immediately severed. This is done by both the PDA and the AMS, and serves as a security measure. It is more difficult to fix a broken situation, than to sever the connection and restart the connection protocol from a clean slate.

CHAPTER 3

Our work

This chapter describes the work that has been done to create Sadako. It is divided into two sections. Section 3.1 describes the initial work that had to be done in order to use the hardware and get Sadako in such a state that it could be experimented with. Those experiments, the findings, and the solutions are described in Section 3.2.

3.1 Initial work to be able to start experimenting

Before the experiments could start, Sadako had to be created. This section describes the process of working with the hardware, writing the client and server software, and creating the communication between them.

3.1.1 On PDA

A lot of work went into molding the Nokia 770 into a workable machine. It is sold as a consumer product, not as a development machine. To have write access to the entire Nokia, it had first to be put into a special developer mode [11]. Only then, the system areas of the machine could be written to.

After having obtained write access, we had to get Java working. The Linux version of Place Lab comes with its own JVM called “j9”, but unfortunately the Place Lab author’s idea of “Linux” is “Linux on a x86 chip”. This meant “j9” was ill suited for the ARM processor of the Nokia. We used a very small JVM called “JamVM” [3]. It is very minimal, but it contained everything we needed to run Place Lab - there is no need for fancy GUIs, at least not at this point.

Place Lab is not entirely written in Java, though. A small part, the part that interfaces with the WiFi card, is written in system-dependent C. In the case of Linux, it uses the Linux Wireless Extensions [12]. To compile this into an ARM library, we had to use Scratchbox [9], a cross-compiling development environment. It can execute ARM binaries transparently on the Nokia 770 while running x86 binaries on the PC Scratchbox is installed onto. This, combined with the compiler toolchains for different processors, makes it a great tool for C development for small devices. Using Scratchbox and quite some sweat, we were able to recompile the native portions of Place Lab into binaries suitable for the Nokia 770.

Once Place Lab was running, it was too slow for experimentation. Startup of Place Lab took over half a minute while the system was doing nothing else. The Java virtual machine had to be started, then Place Lab had to be loaded and executed. As it turned out, it used a HSQL database, which is basically a full SQL server running in the same process as Place Lab. After some research, we found another database interface, in which we could hard-code the access points. Using this method of storing the Access Point information nearly halved the startup time. We did not test this, but the change in database interface most likely also reduced the memory footprint of Sadako.

Place Lab includes a HTTP interface. Through this, other programs can ask Place Lab to do its thing and return a 3D coordinate. We added the interface to the Sadako main class of Place Lab. This way, we could easily let the Python part of Sadako interface with Place Lab. This Python part of the interface is part of the "sadakoclient" Python package. It requests the coordinates via HTTP as described, then converts the returned string into three floating point numbers. That's all there is to it.

The interface with the Sadako server is more complex. It is a RPC client secured with SSL. The RPC protocol was created specifically for this project, with speed and efficiency in mind. It is described in detail in Appendix B. The SadakoRPC client is contained in the "sadakoclient.srpc" Python package. It utilizes Python's dynamic structure and flexibility, which allowed us to create a very nice RPC client in only 72 lines of code [1].

The main loop of the Sadako client is quite simple. It consists of three threads. The main thread connects to the AMS, creates the other two threads, and wait for the user to press Enter. It starts the two threads, and waits for the user to press Enter again. After that, it stops the two threads. The other two threads connect to Place Lab and the Sadako server, respectively. As soon as Place Lab reacts to a measurement request with a set of coordinates, those coordinates are moved from one thread to the other, to be sent over the WiFi connection with the Sadako server. This way, we can take measurements without being slowed down by the transmission to the Sadako server. An overview of the threads can be seen in Figure 3.1.

3.1.2 Communication

There was a lot to be done to get a proper communication protocol. Not only did it have to be secure, it also had to be fast enough to still be able to track a walking person, and it preferably it would have to be a platform-independent protocol.

The Python SSL module of our choice was TLS-Lite. It is a wrapper in itself, and makes the wrapped software more "pythonic". It lacked one thing though, and that was a CRL check. This CRL check is of vital importance to the security of the system; without it, anyone who has ever had access to Sadako will have that access infinitely. It is clear that this is not preferred. Fortunately, the underlying cryptlib [1] module had support for it, so all that was needed was a wrapper that made checking certificates easy and "pythonic".

[1] With all comments, logging and empty lines removed. Including those, the client is 160 lines.

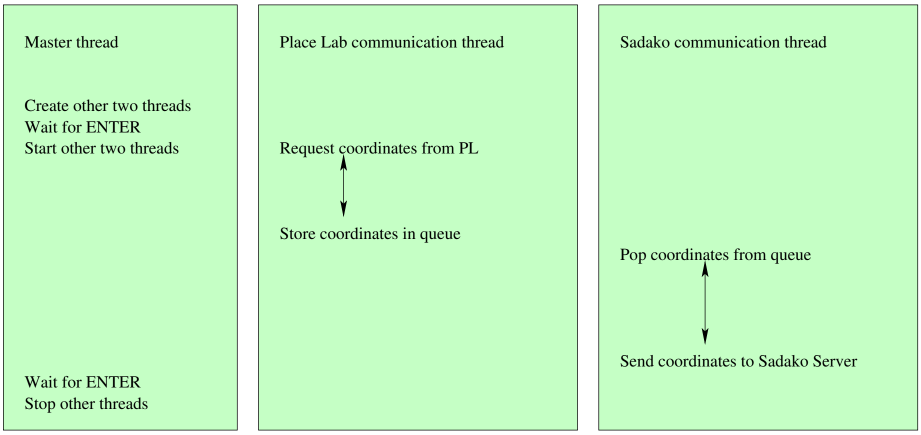
Master thread
Create other two threads
Wait for ENTER
Start other two threads
Wait for ENTER
Stop other threads
Place Lab communication thread
Request coordinates from PL
Store coordinates in queue
Sadako communication thread
Pop coordinates from queue
Send coordinates to Sadako Server

Figure 3.1: Thread overview of the Sadako client

The next step was to wrap Python's XML-RPC server in a SSL connection. When creating the SSL connection, the AMS uses the PDA's fingerprint to look up the connecting user in the database.

At this point, we had a working, secure RPC protocol. What was left was implementing the actual functionality of the AMS. We wrote functions that could retrieve the identity of the connecting PDA, one that could send a new location to the AMS, and one that could retrieve the list of zones the PDA resided in. This last one is mainly for debugging and experimentation, since when everything is implemented, the clients shouldn't care about the different zones - they only care about the alarm being turned off wherever they go.

After implementing the initial versions of the RPC functions, we were able to measure the performance of the communication system to see if it were at all possible to track a walking human. As it turned out, the XML-RPC system as shipped with Python disconnects after receiving the answer to a request, and thus has to reconnect for the next request. This means also performing the SSL handshake and everything else shown in Figure 2.3. The SSL handshake requires quite some computations, which require several seconds on the Nokia's 250 MHz ARM processor. This meant we could do one location update every 4 seconds, assuming estimating the location can be done in parallel with the communication without delaying it. To be able to tell whether this is useful for Sadako, we calculate the distance between two measurements d given a human walking at 4 km/h.

$$\begin{aligned} d &= 4_{km/h} \times 4_s \\ &= 1.111_{m/s} \times 4_s \\ d &= 4.444_m \end{aligned}$$

This means we could do one measurement every 4.4 meters, at best. The measurement itself also requires significant CPU time, since a JVM is not the most CPU-efficient way to run code. All that assumes the owner of the PDA has no intention of using it for anything but Sadako. To overcome this obvious

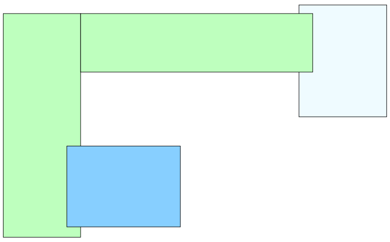

Figure 3.2: Zone information, the green zone is divided into two sub-zones.

issue, we decided to create our own RPC protocol. The requirements for the protocol were:

Persistent connection Only one connection is made, and it is kept open, hence only one costly handshake. Connections are disconnected by the AMS after a certain idle time - 60 seconds at the time of writing. This has the downside that the AMS can only serve about 65000 clients simultaneously, since every client requires an open connection, and there are only that many available TCP ports.

No XML XML is difficult to parse, which is something to consider on a 250 MHz CPU. Instead, we chose a very simple RPC protocol in which each request and each reply is a single ASCII line of at most 200 characters.

No error tolerance When it comes to security, people should not be able to do “funny stuff”. If something is not as described in the protocol description, the connection is immediately severed, and the incident is logged.

This new RPC protocol, which is described in Appendix B, turned out to be much faster than Python’s XML-RPC protocol. It allowed us to perform 8 updates per second, an impressive improvement of 3200%. This meant we could track people, and still have enough time for the CPU to do other tasks.

3.1.3 On AMS

In contrary to the PDA, where we could use the existing Place Lab as a basis, the Alarm Management Server had to be built from scratch.

The Database

First, a database was created to hold information about the alarm zones, the users and their locations. To be able to quickly match a location to a zone, zones are divided into rectangular sub-zones. Figure 3.2 shows three zones, of which one is divided into two sub-zones. The rectangular shape of the sub-zones SZ makes it trivially easy to determine whether a coordinate (x, y) is inside the sub-zone.

$$\text{in zone} = \begin{cases} 1 & \text{if } SZ_{x1} \leq x \leq SZ_{x2} \ \wedge \ SZ_{y1} \leq y \leq SZ_{y2} \\ 0 & \text{otherwise} \end{cases}$$

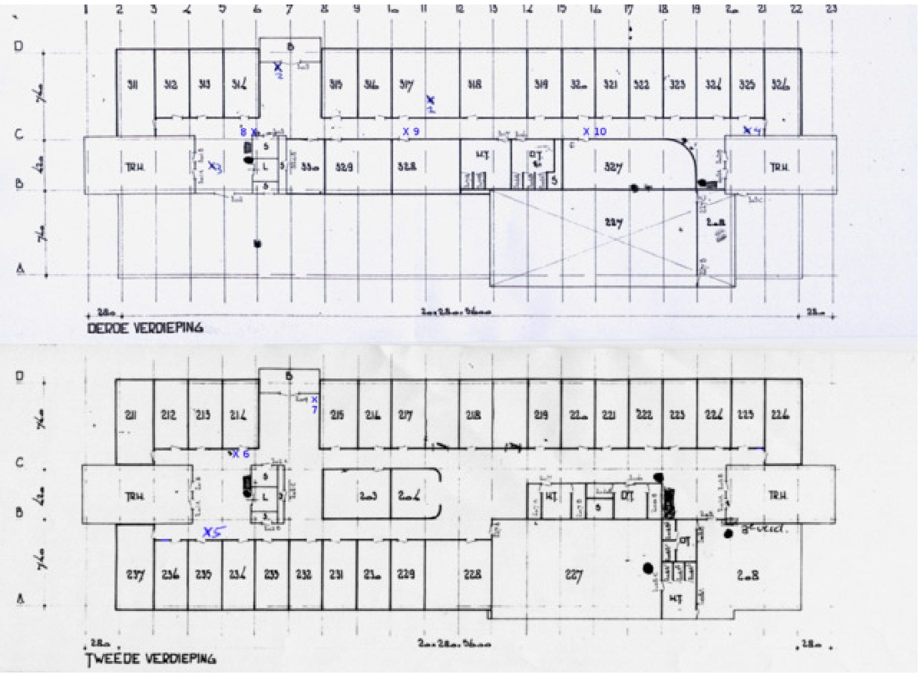

Figure 3.3: The maps of Euclides.

As can be seen in Figure 3.2, zones can also overlap. This ensures that the alarms in both zones are turned off when someone is moving from one zone to the other.

The two floors of the Euclides building were inserted into the database. Because we do not know the real global coordinates of the building[2] we used maps of the floors and used the pixels as a coordinate system. Those maps can be seen in Figure 3.3.

The database also contains user information, such as the user's SSL fingerprint, last known location, whether the user is currently connected or not, and a backlog of the user's locations.

The Graphical User Interface

The GUI displays zones, sub-zones, the locations of the connected users, and the access points. The location of the access points merely serves an informational role during the development of Sadako, and might be removed in future versions.

The interface is split up in several parts, one for each floor. This way, the entire building can be seen in one overview, given that the number of floors actually fit on screen. Figure 3.4 shows an example of the GUI.

The GUI can be used for various purposes. In case of an emergency, a supervisor can quickly get an overview of the location of all registered people in the building. This allows for quick evacuation, and could even save lives.

[2] GPS does not work inside a building

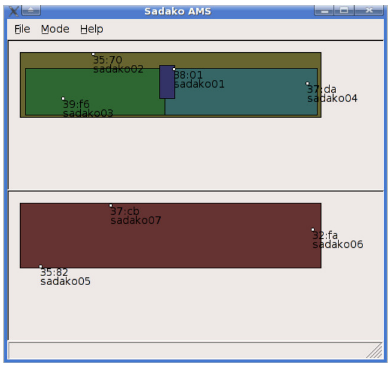

Figure 3.4: The Graphical User Interface. It shows the third (top) and second (bottom) floors of the Euclides building. Nobody is around, hence all zones are dark.

The configuration file

The AMS will have to be suited for different situations. For this reason, we chose to create a configuration file instead of hard-coding settings. An example can be found in Appendix A.2.

Scalability

The scalability of a system is important. The SadakoRPC communication system used has a drawback in this area. Every PDA has a persistent TCP/IP connection to the AMS, hence the number of PDAs a single AMS can handle is limited by the number of TCP/IP ports available on the system. This usually ranges from 32000 to 65000. This allows the system to scale to a large number of users, but it is still limited. With this number of users, handling all the updates could also have a big impact on the performance of the AMS.

To solve this scalability issue, the AMS can be duplicated across multiple computers. Different users can register to different AMSes. The output of the AMS - which zones are used (1) and which are unused (0) - only needs to be filtered through a logical OR filter. This method of duplication allows the different AMSes to have their own database. Even then, the database might be the bottleneck. In that case, load balancing and replication can be used to let a cluster of multiple machines act as a single DBMS.

3.2 Experiments, findings and solutions

After the initial work was done, and we were able to take measurements, the real work could commence. This section describes the most important experiments we have done, including the findings and solutions to issues.

3.2.1 Place Lab always calculated the same coordinates

When Place Lab was first started, and real measurements were possible, we noticed that the calculated coordinates were almost always the same. After investigating the possible causes, we were able to identify three issues that caused this.

The first problem was that the spotter component of Place Lab was unable to initiate a new scan, resulting in old results being presented. This problem was caused by the regular user logged in on the Nokia did not have permission to initiate new scans. This was solved by running Place Lab as root. It does not have to communicate with anything outside the machine it is running on, so this should be relatively safe.

The second problem was that Place Lab did not take signal strength into account at all. The mean of the coordinates of the visible access points was taken as the calculated position of the PDA. This resulted in the same position being calculated whenever the same access points were visible. This was solved by taking the signal strength into account in the calculations, which proved to be quite a delicate issue, and presented various issues in interpreting the signal strength.

The third problem was that the Place Lab code assumed the coordinate system in use were real GPS coordinates. Our coordinate system however, was on a completely different scale. When we moved a few meters, Place Lab thought it was a few *thousand* meters. It remembered the last calculated position, and all access points further than 5 km away from that position were ignored in the next measurement. Due to our different coordinate system, all but the nearest access points were more than 5 km away according to Place Lab, hence they were all ignored. This was solved by removing the 5 km limit.

3.2.2 Coordinates always calculated in convex hull

The coordinates are always calculated inside the convex hull around the access points - see Section 2.2. This means that the access points have to be spread across the edges of the building, so the inside can be properly scanned. Experiments showed that the coordinates calculated were not only inside the convex hull, they were almost near the center of the hull. Due to the heterogeneous nature of the PDAs that should be able to run with Sadako, the maximum RSS measured when standing straight under an AP is unknown. The relation of signal strength and environmental changes such as changing temperature, humidity, and air pressure only add to this uncertainty. This means we never calculate a zero distance between PDA and AP, hence the calculated coordinates are always closer to the center of the convex hull.

Later on, we describe another system to measure the location, which solved the "we always measure very near the center" issue, and gave us a much more realistic spread of the measured coordinates.

3.2.3 Distorted "view of reality"

Place Lab has a distorted "view of reality", caused by walls and other signal blocking and deflecting objects. When going around a corner and coming into more direct view of an AP, the PDA is measured closer to that AP due to a

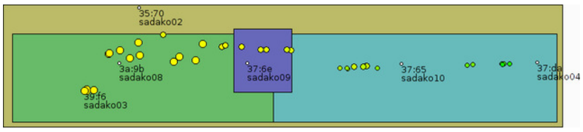

Figure 3.5: Replay mode

stronger received signal. Can be solved by taking measurements and mapping the measured coordinates to the real coordinates, but this is probably not required. This distortion only causes a shift of a few meters. With zone sizes being several orders of magnitude larger than this distortion, it is not a real issue for the Sadako project. If more fine-grained positioning is required, this mapping from measured to real coordinates could provide the solution.

3.2.4 Cannot use AMS GUI and walk with PDA simultaneously

For experiments where only a single person is working on the project, like during the development of Sadako, it is physically not feasible to walk more than a few meters and still have a clear look at the AMS GUI. To solve this issue, we implemented a “replay” function. Press CTRL+R in the GUI to activate “replay” mode. An example can be seen in Figure 3.5.

All measured locations are stored on the AMS. Those locations are grouped by timestamp. If the timestamps of two location measurements are more than a minute apart, they belong to two different groups. Activating the “replay” mode displays the last group that was recorded, showing all recorded locations simultaneously, colour coded by their relative time. The oldest location is yellow and large, while the latest location is green and small.

The replay function currently only works if there is a single person walking around. It was made especially for the purpose of allowing solo experiments. Of course, it could be augmented to filter out a single person, or to give different people different colours.

3.2.5 The distance function

In Section 2.1 we showed the distance function that was empirically found on April 27th, 2006. A week later, at May 5th, we performed the same experiment, and combined the results into Figure 3.6. As can be clearly seen, this meant the distance function we originally found was unusable. During the second experiment, the temperature was much higher, and probably the air pressure was different too, compared to the first experiment. If we were able to measure the angle of the AP’s signal we could obtain a better precision. Unfortunately, that is not possible given the required 802.11 hardware.

Due to the above reasons, we dropped the idea of a distance function. The fluctuation in RSS is simply too large to determine distance to an access point based on a single measurement. Better estimates are possible, but require multiple measurements from the same position, which is impossible while walking. The finally used method of determining the position can be described as:

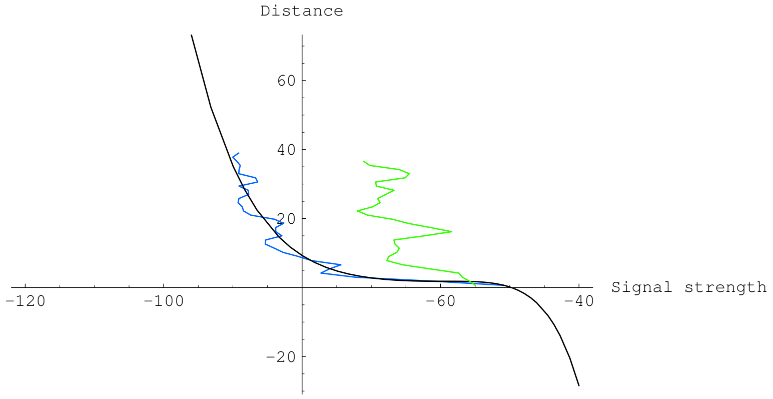

Figure 3.6: Measured signal strength in dBm against distance in meters, both from April 26th (blue) and May 5th (green). Every point is the average of 30 measurements.

1. Find maximum M and minimum m RSS of all beacons in this measurement. Using those values causes an independence on the absolute values of the readings. This masks atmospheric changes and other forms of interference.

2. Determine weight w_i of access point i based on its RSS r_i:

$$w_i = \left(\frac{r_i - m}{M - m}\right)^2$$

   The inner part of this function, $\frac{M-r_i}{M-m}$, causes a linear spread of the weights. The strongest AP will get weight 1, the weakest weight 0. All other access points are linearly spread between those values. The squaring of those weights gives even more relative importance to the strongest APs as it gives less to the weaker APs: $0.5^2 = 0.25$ but $0.99^2 \approx 0.98$.

3. Determine elevation $\mathbf{p}_z$ of PDA by using the elevation of the visible access points $\mathbf{AP_i}_z$:

$$\mathbf{p}_z = \frac{\sum w_i \mathbf{AP_i}_z}{\sum w_i}$$

   This is simply a weighed average, where the weights are normalized by dividing by their sum.

4. Determine horizontal position $\mathbf{p}_{xy}$ of PDA by using the horizontal position $\mathbf{AP_i}_{xy}$ of all visible access points on the same floor:

$$\mathbf{p}_{xy} = \frac{\sum_{\textit{on same floor}} w_i \mathbf{AP_i}_{xy}}{\sum_{\textit{on same floor}} w_i}$$

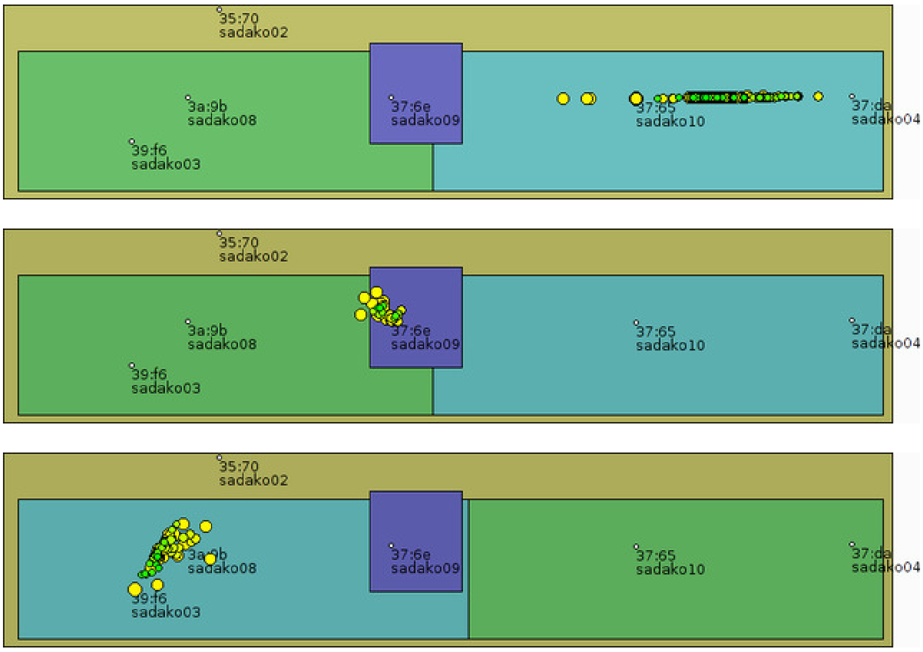

Figure 3.7: Measurement noise when idle.

With just one or two access points visible, this method uses the coordinates of the strongest received AP as **p**. Every extra visible AP will improve the measurements. For example, if APs A, B and C are visible, and A and B measure at -66 dBm, while C measures at -80 dBm, the calculated position will be between A and B.

The thick concrete between floors makes it harder to use APs from different floors determine the position. This is why all visible access points are used to determine on which floor the user is located, and then only the APs on that floor are used to determine the horizontal position.

To illustrate the measuring noise, and the effect of the environment, we performed three series of measurements. The PDA was placed on the ground or another supporting surface, and measurements were taken during approximately a minute. Figure 3.7 shows the results of the three series. The first picture shows a lot of noise due to the metal cupboards under and next to the PDA. This noise was amplified by the fact that the PDA was located between two APs. The second picture was taken with the PDA on a metal desk, but not surrounded by metal, and close to an AP. The third image shows the PDA laying on the floor between APs, demonstrating the convex hull effect discussed in Section 3.2.2 – the measurements do not cross the line between *sadako02* and *sadako03*. An important observation is that in spite of the noise the measured locations are all within the same zone.

3.2.6 Access point placement

The placement of the access points is very important for the functioning of the Sadako system. As described in the previous section, creating a distance function solely based on signal strength is not possible given the used hardware. To still obtain a usable precision when determining the position of Sadako users, we increased the number of access points, and decreased their signal strength. This improves the precision of the measurements, similar as when Bluetooth hardware is used [14].

The lower signal strength, the higher AP density, and the weighing method described in the previous section combined gave us a stable, usable way of determining the location of the Sadako users. This solved the issue presented in Section 3.2.2, and also gave us a much cleaner, less constant-ridden function than the one found in Section 2.1.

Instead of placing the access points at the edge of the building, as was required by the now obsolete method, the new method works best if the APs are placed in places of interest, and at most 8 meters apart. Interesting places include the centers of zones, stairways, and ends of corridors.

3.2.7 Noise causes single readings on another floor

Sometimes, noise causes the Sadako system to measure the PDA on a different floor. For example, if Alice is measured on the third floor for 30 seconds, then one measurement claims she is on the second floor, and a subsequent measurement tells the AMS she is back on the third floor, the measurement at the second floor most likely was a fluke, and should be filtered. The Place Lab code is altered to accomplish this. If $\mathbf{p}'$ is her previous position, the elevation is now calculated as:

$$\mathbf{p}_z \quad = \quad \frac{\mathbf{p}'_z + \sum w_i \mathbf{AP_i}_z}{2 \sum w_i}$$

This formula is simple yet effective.

CHAPTER 4

Security and Privacy

The privacy of the users of Sadako is important. Place Lab has been built with privacy as one of its goals [18]. Other systems, like Active Badge, Active Bats and PARCTab use an infrastructure that consists of receivers deployed in places of interest, with end-users beaconing out data stating that "I am here". In contrast, Place Lab and other systems consist of beacons deployed in places of interest, signaling to end-users that "You are here" [18]. The latter approach has the advantage that the location of the PDA is not broadcast.

4.1 Communication

Unfortunately, Sadako has to break with this form of privacy, due to the simple reason that communication with the AMS is required. This communication can be seen by others, just as the APs can be seen by Place Lab. However, this communication is encrypted, so the contents cannot be read within a reasonable amount of time[1]

4.1.1 TLS/SSL

The security of the transport layer is managed by TLS/SSL. This takes care of authentication of both parties, via methods described in Section 2.3. The encryption function itself is determined by the list of functions supported by both parties. In the case of the Sadako development environment, this is an 256-bit AES encryption. AES is the Advanced Encryption Standard, the successor of DES, and is also known as Rijndael [19] encryption. At the moment of this writing, there is no known faster method of breaking this encryption than exhaustive key search. None of the recently published ideas has lead to an attack [15][2]. Given that there are 2^{256} possible keys, this presents a tremendous search space, and makes the encryption good enough for our use.

[1] Decrypting the data would take years if not centuries, after which it probably is of no interest to anybody to know at which time someone went to the toilet.

[2] Something is considered an attack if it is faster than an exhaustive key search.

4.1.2 RPC

All the RPC functions receive the client information from the database. This information is retrieved using the identification obtained from the TLS/SSL handshake. Unless an imposter has access to the victim's private key, this information is virtually impossible to fake.

The client information is looked up by her SSL certificate's SHA1 fingerprint. There is a very small chance that two generated SSL certificates share the same fingerprint, and thus would be indistinguishable from each other. Fortunately, the database requires that each user has a unique fingerprint. This prevents such attacks.

Many attacks are caused by buffer overruns. Those are caused by sending more data than anticipated, if the receiving end of the data does not implement a proper check on the length of the received data. By disconnecting the client when too much data is received, this kind of attack is impossible.

4.2 Data storage

The location data is stored in a central database. At this moment, this database is unencrypted. There is always the question of who has access to this data, and what will be done with it.

At least the data should be temporarily stored. For the working of Sadako, only the current location of users is required, and not the entire backlog – except for the replay function, which is only useful during development. This means that, if no other requirement arises, the backlog is secure because it is gone.

If there is the requirement of a backlog for security or review reasons, the backlog is as secure as the machine itself. In such a case, extra precautions could be taken, for instance by logging the movements to another system that is not publicly reachable. For optimal security, this could be a system that is connected via a serial cable, which receives only (*identifier*, *coordinate*, *timestamp*) tuples over that cable. In such a setup, the connection it does have can not be used to gain access to the system[3].

Since physical access to a machine means it can be broken into, it goes without saying that the AMS and the backlog database machine need to be physically secured.

If the backlog is secure, there is still the current location of the user. If security is breached, this could be tracked by an attacker, allowing the attacker to create his own backlog. This issue is not directly solvable, since the GUI displays users at their current location, hence that location needs to be known. A possible alteration would be to only store the current zones a user is in, and let the GUI only display the number of users in a certain zone. Location information could then still be leaked in the same fashion, but it would reveal less detailed and thus less privacy-sensitive information. Of course, the measured location is still at some point known to the AMS, and thus can be captured by the attacker. Since it would no longer be stored in the database, however, the attacker would need to be able to read and interpret Python's memory. Due to the dynamic

[3] Assuming there are no buffer overflows in the part that receives and parses said tuples, which is very likely if it is written in a dynamic language like Python

nature of Python, the memory locations used to store received PDA locations change all the time, which would make this a very difficult attack vector.

CHAPTER 5

Future improvements

This chapter will highlight some ideas we have to improve upon Sadako, but fall outside the limits of this project. Of course, some have already been mentioned and will not be repeated here.

Different users could get different colours, but discussion is needed to determine if this is acceptable or not. Being able to uniquely identify users directly from the AMS GUI might be interpreted as a breach of privacy.

The PDA currently has no GUI, only a minimal text interface. Before using Sadako as a commercial product, it would at least require a graphical interface, perhaps combined with sounds to alert the user about certain events. Of course, a nice installer would also be very welcome.

The entire Sadako system is in English. For better adoption, it should be internationalized and translated. This is more important for the PDA than for the AMS, since the PDA will be used by a wider range of people.

At the start of the project, we discussed linking the measurement frequency to the user's speed. At low speeds, the frequency could be lowered to save CPU and battery power. Even though the idea is still good, we are unsure about the feasibility, since noise in the measurements makes it hard to determine whether a user is idle.

Currently, when a zone is empty, the alarm in that zone will immediately switch on. Due to noisy measurements, this might cause false alarms in cases where a person is in zone A but measured in zone B. To prevent this, and to reduce the wear and tear on the alarm system relays, a timeout could be introduced. With this, the alarm is turned on only if the zone has been empty for T seconds, and of course turned off immediately when the zone becomes occupied.

More information could be obtained from the database. At this moment, the location and identification of the access points is stored in the database purely for the GUI, but of course those could be fetched by the PDA and used for positioning too. That would make updating the access point database on the PDAs a trivial matter. Besides this, the coordinates at which one floor ends and another starts are hard-coded. This could very well be included in the database as well. It would certainly ease the installation process.

CHAPTER 6

Conclusions

Sadako provides an easy to use alarm management system. By using properly placed access points, it could even become more than just that – it could be used to guide visitors to certain locations, for example.

At this moment the Sadako prototype has not been linked to a real alarm system. Creating such a connection requires a specific interface to the hardware system, since every alarm system will have to be controlled in different ways. Due to the simple output of the Sadako AMS - one function to turn the alarm on, one to turn it off - this should pose little problems to someone with experience in computer-controlled alarm servers.

There are multiple systems that use radio signals for location and identification of people. One of the most well known is RFID, Radio Frequency Identifier, with uses ranging from measuring start/finish times in Formula 1 racing, to identifying your cat and opening the cat flap. All of those systems, however, use specialized hardware. Other systems like Active Batch, Active Bats and PARCTab broadcast the user's location in a way that anyone can hear. As far as we have been able to tell, nobody has hooked up common IEEE 802.11 hardware in such a way that a privacy-observant alarm management system like Sadako is created.

In the future, Sadako might become a platform that is compatible with a mixture of common and custom hardware. The APs now in use could be replaced by cheaper, custom hardware that send out IEEE 802.11 compatible radio signals. The code of the Sadako client could be rewritten into Java, integrated with Place Lab into a single application, and run on an embedded 802.11-enabled Java chip. Combined with a WiFi chip, this could form a small Sadako badge. Users would then have the choice of using the Sadako badge or their own PDA, since the radio signals will be compatible with both.

Bibliography

[1] Cryptlib Encryption Toolkit.
http://www.cs.auckland.ac.nz/ pgut001/cryptlib/.

[2] Host AP driver for Intersil Prism2/2.5/3, hostapd, and WPA Supplicant.
http://hostap.epitest.fi/.

[3] JamVM. JamVM is a Java Virtual Machine which conforms to the JVM specification version 2 (blue book). In comparison to most other VM's (free and commercial) it is extremely small.
http://jamvm.sourceforge.net/.

[4] Maemo. Maemo is a development platform to create applications for the Nokia 770 Internet Tablet and other maemo compliant handheld devices.
http://www.maemo.org/.

[5] Nokia 770 Internet Tablet.
http://www.nokia.nl/uk/Phones/PhoneModels/770/.

[6] OpenSSL. A popular Open Source SSL implementation, created by the OpenBSD team.
http://www.openssl.org/.

[7] Place Lab. A privacy-observant location system.
http://www.placelab.org/.

[8] PostgreSQL relational database.
http://www.postgresql.org/.

[9] Scratchbox. Scratchbox is a cross-compilation toolkit designed to make embedded Linux application development easier.
http://www.scratchbox.org/.

[10] Ubuntu Linux.
http://www.ubuntulinux.org/.

[11] Using flasher and the reference root filesystem.
http://www.maemo.org/platform/docs/howtos/howto_use_flasher_rootfs.html.

[12] Wireless Tools for Linux.
http://www.hpl.hp.co.uk/personal/Jean_Tourrilhes/Linux/Tools.html.

[13] Paul C. Kocher Alan O. Freier, Philip Karlton. The SSL Protocol, version 3.0. 18 November 1996.
http://wp.netscape.com/eng/ssl3/draft302.txt.

[14] Sunny Consolvo Anthony LaMarca, Yatin Chawathe. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Proceedings of Pervasive 2005, Munich, Germany*, 2005.

[15] V Rijmen E Oswald, J Daemen. AES-The State of the Art of Rijndael's Security. 30 October 2002.

[16] Guido van Rossum. The Python programming language.
http://www.python.org/.

[17] Institute of Electrical and Electronics Engineers. IEEE 802.11(tm) Wireless Local Area Networks. Technical report, IEEE.
http://grouper.ieee.org/groups/802/11/.

[18] James A. Landay Jason I. Hong, Gaetano Boriello. Privacy and Security in the Location-enhanced World Wide Web. In *Ubicomp 2003, Seattle, WA, United States of America*, October 2003.

[19] Vincent Rijmen Joan Daemen. *The Design of Rijndael*. Springer Verlag, 2002.

[20] Sun Microsystems. The Java programming language.
http://java.sun.com/.

[21] Hans-Jürgen Zepernick Tadeusz A. Wysocki. Characterization of the indoor radio propagation channel at 2.4 GHz. *Journal of telecommunications and information technology*, 4 March 2000.

[22] Triangulation. Triangulation.
http://en.wikipedia.org/wiki/Triangulation.

APPENDIX A

Software manual

This chapter describes the commands and actions required to get Sadako up and running. It also describes the commands and actions for updating the Sadako software on the Nokia 770 Internet Tablet, in case further development is done.

A.1 Running Sadako

To get Sadako working in the development environment, follow these steps.

1. Start the AMS PC.
2. Start all access points by plugging in the power supplies.
3. Start the AMS software:
   - cd sadako/ams
   - ./ams
4. Start the Nokia 770.
5. Connect the WiFi card of the Nokia with the Sadako access point connected to the development machine.
6. SSH to the nokia and start Place Lab as root:
   - ssh user@nokia770
   - sudo su -
   - mount /media/mmc1
   - cd placelab/run
   - ./sadako
7. Open a terminal on the Nokia and start the Sadako client:
   - cd sadako/pda
   - ./sadako
8. Walk around, and look at the AMS display. The Nokia will display the zones you are in, every time they change.

A.2 Configuring the AMS

.

The following is an example of the Alarm Management Server configuration file. Logging channels that are not configured will be automatically added to the file with the “debug” log level.

```
[SSL]
certpath = /home/sybren/sadako/ssl-test
servercert = %%(certpath)s/testserver.crt
serverkey = %%(certpath)s/testserver.key
cacert = %%(certpath)s/unrealtower-ca.crt
crl = %%(certpath)s/crl

[logger]
sadako = debug
sadako.config = info
sadako.db.dbdef = info
sadako.db = info
sadako.gui = debug
sadako.srpc = info
sadako.ssl = info
sadako.xmlrpc = info

[PostgreSQL]
username = sadako
password = SECRET
hostname = localhost
database = sadako
```

A.3 Configuring the PDA

The following is the configuration part of the Sadako Client configuration file, as stored in “sadakoclient.settings”:

```
# The directory containing all certificates
certpath = '%s/sadako/pda' % os.environ['HOME']

# Server fingerprint
server_fp = '11995569A6975D6B8334E43C31FFFAA645908F1E'

# Server URL
server_url = 'sadako://172.17.0.2:8000'

# Client certificate
client_crt = file(os.path.join(certpath, 'client.crt'))

# Client key
client_key = file(os.path.join(certpath, 'client.key'))
```

```
# CA certificate
ca_crt = file(os.path.join(certpath, 'unrealtower-ca.crt'))

# Place lab proxy port
pl_proxy_port = 2080

# Place lab IP address
pl_address = '127.0.0.1'
```

A.4 Updating Sadako on the PDA

There are two parts, the Python Sadako Client and the Sadako build of Place Lab.

A.4.1 Python Sadako Client

Ensure the Nokia 770 is associated with the Sadako access point connected to the development machine. On the development machine, type:

- cd sadako/pda
- make publish

A.4.2 Sadako build of Place Lab

Ensure the Nokia 770 is associated with the Sadako access point connected to the development machine. On the development machine, type:

- cd placelab/src
- make

APPENDIX B
SadakoRPC

The SadakoRPC protocol uses TCP/IP port number 8000. It uses SSL 3.0 and TLS 1.0, and is compatible with the "openssl s_client" command. Commands and replies consist of a single ASCII line of at most 200 characters.

Request The words making up the line are separated by a single space. The first word is the function to call, the other words are the arguments. Only simple arguments are supported, like strings, integers and floating point numbers.

Reply The reply is a free-form string depending on the function called.

The SadakoRPC server process has to pass an object to the SadakoRPC module. All functions in this object that do not start with an underscore are exposed as RPC functions. Those functions will be called with an extra argument after the obligatory "self" parameter. This extra argument contains the client information from the database, so that the calling function "knows" who it's talking to. The other arguments of the called function will be strings, unless the "sadako.srpc.arguments" function decorator is used to cast the strings to something else.

When a client disconnects, the "_disconnect" function is called, so that the server application can also properly handle disconnects. The Sadako AMS uses this for instance to update the GUI and alarm zone status.

B.1 The built-in SadakoRPC functions

Function	**Args**	**Description**
BYE	-	Properly disconnects the SadakoRPC connection. Returns the string "GOODBYE".
FUNCTIONLIST	-	Returns a list of functions exposed through SadakoRPC, as a space separated list. This is used by the client to reject calls to non-existing functions.

B.2 The Sadako-specific SadakoRPC functions

Function	**Args**	**Description**
identify	-	Returns the client's SSL SHA1 fingerprint
location	-	Returns the location in the form (x, y, z)
update_location	x y z	Sends location (x, y, z) to the server. This updates the client's location in the database. Returns the stored location, which might be different than the sent location due to interpolation and guestimation.
zones	-	Returns the list of zones the client is currently in, as a Python string representation of a list.

APPENDIX C

Access point information

For the Sadako development and experiments, ten Linksys WAP55AG access points were used. Configuration of the access points, for $01 \leq N \leq 10$. Settings that are not changed from their default setting are not mentioned. Logging in on the web interface of the access points requires authentication. The username should be empty, and the password is specified below.

Setup **Device name:** sadako$N$

Configuration type: Automatic/DHCP

Basic wireless **Wireless-A:** Disabled

Wireless-A country: Netherlands

Wireless-G: Enabled

Wireless-G Mode: Automatic

Wireless-G SSID: sadako$N$

Wireless-G Channel: N

Wireless-G SSID broadcast: Enabled

Advanced Wireless G Settings **Transmission power:** See Section 3.2.6

Administration **Management password:** I'm not your mommy!