Semester Thesis "BlueLocation"

Katrin Bretscher

SA-2005-17
Summer Semester 2005
Supervisor: Prof. Dr. Plattner
Advisors: Vincent Lenders, Matthias Bossardt, Martin May

Abstract

This thesis describes the design and implementation of a system to locate mobile users by using their phones' Bluetooth radio. In addition, it proposes a user application that allows a user to get information about the current location of another user.

Diese Semesterarbeit beschreibt den Entwurf und die Implementierung eines Systemes, welches mobile Benutzer lokalisiert. Dazu benuetzt es den Bluetooth-Funk der Mobiltelefone der Benutzer. Zusaetzlich beschreibt diese Arbeit eine Anwendug fuer Mobiltelefone, welche es Benutzern erlaubt, Informationen ueber den Aufenthaltsort eines anderen Benutzers zu erhalten.

Contents

Chapter 1

Introduction

1.1 Motivation

Imagine yourself at a ball. A large crowd, good music, you're in the mood of dancing - and you can not find your partner. At large events it often happens that people are not able to find their friends. Calling them on their mobile phones is an option, but the noise at such events makes it difficult to hear a person talk, or a phone ring. It would be much easier if the phone was able to tell a person's friends where its owner is without him having to do anything.

Figure 1.1: How do you find your partner?

This requires that the phones are able to find out information about their location. One option is to request location information from a system such as GPS [1] or the mobile phone provider. But as events often take place in rather small places and on more than one level of a building, the information provided by these services is not useful for our task: We need detailed information such as in which room a phone is located and what the name of that room is: A person attending a ball should be directed to the Waltz hall rather than 35 meters northeast from her current location. Such information can only be provided by a system run by the event's own infrastructure.

When we have a system that is able to find out where a person currently is and that can store that data, we can consider other uses of our system: We can compute statistics and establish mobility patterns. How long does a person stay in one room? How many rooms do people visit on one evening? Are there groups of people that behave in the same way? Knowing about the users' behavior is important for many people: Event organisers can place foodstands, ticket booths, and sponsors' advertisements in places where the visitors are most likely to pass by. Security guards learn about peoples' behavior and can place guards, paramedics, or fire exstinguishers at the most frequented places. In case of an emergency, they know how many people are in which part of the building. Scientists designing buildings or working with mobile components no longer have to assume that users move randomly, but can rely on the data gathered in a real environment.

If we want to use people's phones in that system, we have to take into consideration that most people do not have the latest model of mobile phone and we thus might not be able to use the most modern technologies. Most phones support a variety of interfaces such as Infrared ports and Bluetooth [2]. Bluetooth is the optimal choice of technology for the system described as its usage is free, and its range is of the desired granularity.

Finding information about a phone's location is not the only application for Bluetooth: As with most technologies, its inventors and sellers claim that one can do anything and everything with it. To find out what one could want to do with Bluetooth enabled devices and which of those ideas are feasible, the Computer Engineering and Networks Laboratory (TIK) at the Federal Institute of Technology (ETH) in Zurich, Switzerland, has founded the Blue* [BlueStar] [3] project. It serves as a platform for all applications for Bluetooth enabled devices. As such, this thesis is part of the Blue* project, and hence also its name: "BlueLocation".

1.2 Task Description

Before we can help two dancers find each other, we must find out where they are. We will store this information in a central unit, and when we want to tell someone where her friend is, we will query that unit.

The task description is summarized as follows:

> The goal of this thesis is to develop a locating infrastructure for indoor purposes. Such an infrastructure consists of a certain number of fixed Bluetooth nodes to determine connectivity of mobile users, and a central instance that collects and keeps track of position information.

For more detailed information, please refer to the full task description in appendix A.

1.3 Overview

This document is organised as follows: The following chapter gives an overview over related systems and theses in the Blue* project. We then explain which restrictions this imposed in chapter 3. Chapter 4 discusses security issues and design alternatives, and chapter 5 describes the implementation thereof. In Chapter 6, we propose future extensions to the system, and conclude.

Chapter 2

Related Work

2.1 Other Blue* Projects

Bluetella

Bluetella is a peer to peer file sharing application designed for Bluetooth enabled mobile phones. It was originally designed in 2003 by Ganymed Stanek [4] on simulators only. Andreas Weibel and Lukas Winterhalter redesigned and reimplemented Bluetella in 2005 [5] on real mobile phones, when the Bluetooth technology was finally available for these devices.

BlueDating

In parallel to the reimplementation of Bluetella, Sandro Schifferle and Christian Braun implemented a dating application for Bluetooth enabled mobile phones [6]. Similar to Bluetella, BlueDating is based on the peer to peer paradigm. It allows its user to give details about himself such as color of hair, height, or interests, and searches for other persons in Bluetooth range that are looking for a user with the given properties.

BlueStar

Many applications involving Bluetooth enabled mobile phones tend to have a similar architecture. For this reason, Nicole Hatt created a framework [7] for the Blue* project. Her work was not finished at the time this thesis was written, and so unfortunately, we could not build upon it.

2.2 Multiuser Bluetooth Applications

Bluetooth enabled mobile phones can interact with any other such device. In most cases, no other person but the phone's owner is involved in the actions taken: Bluetooth is used to establish a connection between the phone and a headset or to send a picture from the phone to a desktop computer. Applications for interaction with other people are rather rare. Most of them, such as BuZZone [8], mobiLuck [9], or proxidating [10], are applications to find people within Bluetooth range. They search for other Bluetooth devices that offer the same service, search for people matching a given profile and allow the user to exchange messages. Some of those systems, such as nTag [11], spotMe [12], or Speck [13] even run on dedicated devices. Those devices are handed out to visitors at special events such as conferences or exhibitions.

All of those applications are limited in the way that they only allow users to search for other users within Bluetooth range. Most of them do not allow the user to actively search for another user.

Chapter 3

Environmental Constraints

3.1 Available Hardware

As explained in the introduction, this thesis is one in a series of theses being written about Bluetooth enabled mobile phones in the course of the Blue* project. For economical and practical reasons, all these theses are implemented on the same phones: The second Bluetella thesis [5] evaluated several mobile phones for their usability for the given tasks, and suggests to use Nokia 6630 mobile phones. These phones were also the phones used for this thesis.

Figure 3.1: Nokia 6630 Bluetooth enabled mobile phone

The central unit and the fixed Bluetooth nodes are a peculiarity of the system to be built, and therefore there were no restrictions on hardware to be used for this purpose.

3.2 Technology Restrictions

The introduction shortly describes why Bluetooth is the optimal choice of technology for interaction with the users' mobile phones: The phones used by most people are Bluetooth enabled, and Bluetooth has (other than the ever more widespread Infrared ports) the desired range of roughly room size. Having a smaller range, for example only one meter, would mean that either a lot of phones pass our system unnoticed, or we have to put a fixed node that does take notice of the phone every meter. If we have a bigger granularity than Bluetooth, such as WLAN, we would not be able to locate a person accurately enough: A phone communicating with a fixed node might be anywhere within the range of the fixed node, including a story above or beneath the fixed node. While it is theoretically possible to reason about the distance of another device based on strength, angle, and delay of a signal, this is not feasible in our case: There is no programmatic approach to find out details about a signal received on a phone for third-party programmers as we are.

Communication Technologies

Bluetooth is optimal for us as we can assume that a phone is not too far away from a fixed node receiving the signal, and yet we can also assume that one fixed node per room is sufficient to receive a signal from all phones in that room.

Having Bluetooth as the main communication means for the system also is the reason why this thesis is part of the Blue* project.

Requiring the fixed nodes to communicate with the phones via Bluetooth does not limit other communication channels to any technologies. Fixed nodes can communicate with each other and the central unit with any technology that is most feasible in the given circumstances. The next chapter discusses possible alternatives for the non-phone communication.

Programming Languages and Platforms

The phones available for this thesis are programmable in Java and Python. As the support for Python was only made available shortly before this thesis was started, we decided to go with the proved and tested option, and chose Java as a programming language for the phones.

We were completely free to choose any hardware we found suitable for fixed nodes and central unit, and thus also had no restrictions on which platform or programming languages to use.

3.3 Human Resources

This thesis was written as part of the Computer Science curriculum at the Federal Institute of Technology. For such, the estimated effort is roughly 150 work hours per person. Originally, the thesis as described in the task description in Appendix A was issued for Daniel Hottinger and Katrin Bretscher, but Daniel left the project for personal reasons before any work had been done.

Chapter 4

Design

4.1 Overall Architecture

While the BlueLocation system is running, we want to quickly retrieve and calculate data. We also want to store the data permanently in order to be able to evaluate it even after the system has been shut down: One purpose of the BlueLocation system is to provide the data for computing statistics about mobile users' behavior. It is not feasible to simply store the data at the fixed node where it is gathered: We possibly need to access any data from any part in the system. For this, we need to know where in the system the desired data is stored. The easiest way of implementing this is to have a central unit that stores all data, and allows for fast access, retrieval, and calculating of data. Such a system is commonly known as a database. The database serves as the *model* of the system.

When we want to respond to requests for data, we need a component that is able to accept, verify, and process such requests, and to compute and send an according response. This corresponds to the fixed nodes which initially accept a request by a phone, and the central unit which verifies the request and responds via the fixed nodes. The fixed nodes and the central unit's logic thus play the role of the *controller* of the system.

Via those controllers, the users' phones can request certain data to show to the user. The information about whom she can search, who can search for her, and where her friends are make up this user's *view* of the BlueLocation system.

4.2 Design Alternatives

4.2.1 Controller Logic Distribution

As described in the previous section, the central unit and the fixed nodes make up the controlling part of the system. Up to now, we have not made any assumptions about which part of this logic will be computed by the central unit, and which by the fixed nodes. How much logic we want to source out to the fixed nodes depends on how up-to-date the data at the central unit and the fixed nodes needs to be, how much load we can put on the network, and on the computing power of the hardware.

As we were free to choose any hardware for fixed nodes and central unit, we first evaluated design alternatives and their impact on hardware requirements, and then decided on which devices to use.

In the following, we discuss the two main alternatives for distributing controller logic: One where the fixed node does not perform any controller logic computation and one where the fixed

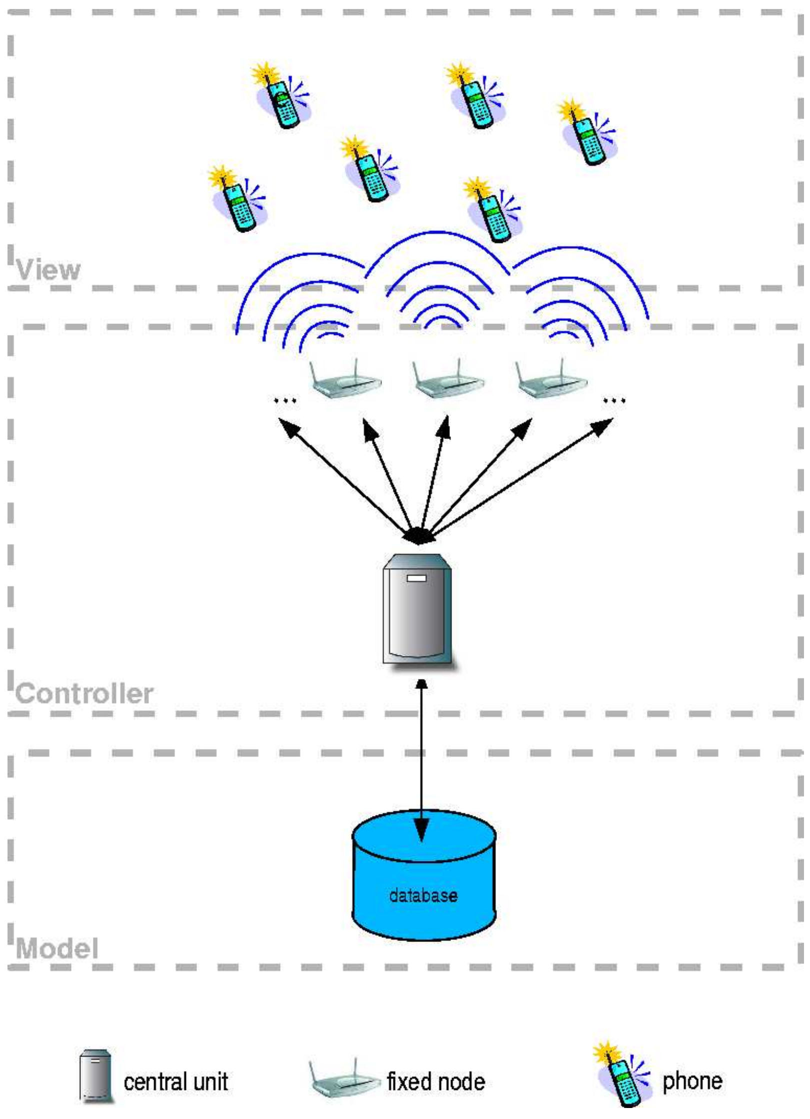


Figure 4.1: Model-View-Controller architecture

node performs an arbitrary amount of computation.

Forwarding Fixed Nodes

If we assume that both the network and the central unit are powerful enough to handle all messages generated by user requests, the central unit can compute everything. Fixed Nodes then are merely its remote branches: They simply forward all requests by the mobile phones to the central unit and vice versa. Their functionality can be viewed as being on the transport layer of the communication rather than on the application layer.

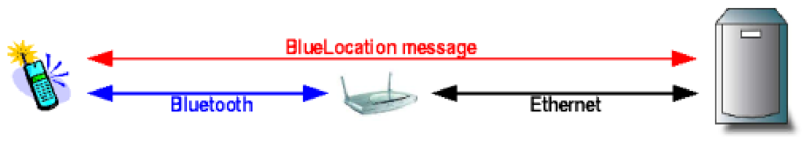


Figure 4.2: fixed nodes forward all messages

Fixed nodes for this type of connection are very cheap both for buying and for maintaining. Programming them is reduced to configuring the device, or, at the most, writing a very simple program.

Since in this design, the fixed nodes do not change anything about the messages exchanged, it is necessary that the phones are able to communicate with the central unit directly. Technically speaking, the phone has to have a TCP/IP connection to the central unit. This connection is built on Bluetooth from the phone to the fixed node, and on Ethernet or WLAN from the fixed node the the central unit.

In this scenario, the phone first must perform a Bluetooth inquiry[1] to find a fixed node, then negotiate an IP address with a DHCP server before it can send its request (via TCP/IP) to the central unit. A phone that is able to communicate with the IP protocol over Bluetooth must implement the Bluetooth LAN profile[2]. While there are phones that do so[3], the Nokia 6630 phones available for this thesis do not provide this profile.

Besides for the Bluetooth LAN profile not being available on most mobile phones, this design has other drawbacks:

- As explained above, the setup for exchanging a message includes both a Bluetooth inquiry and a DHCP negotiation phase. During that time, the user should not move too far for the connection to be stable. Unfortunately, that is what users normally do, and they will lose the connection before their request can be answered.
- Having a TCP/IP connection to the central unit requires that the phone's application knows about the location - the IP address - of the central unit. Either the user enters this manually every time she enters a location that provides a BlueLocation system, or she downloads a new BlueLocation application to her phone for every BlueLocation event she visits.
- The phone actively has to communicate with the central unit to update the data about its location. This requires resources which some of the popular phones might not have: Running the application may leave the phones reacting slowly to user interaction, and the user will most probably shut the application down. If the fixed nodes themselves were able to see

[1]A Bluetooth inquiry is a process that searches for devices in range that have Bluetooth turned on. It is needed as a first step before one can send a message via Bluetooth

[2]A Bluetooth profile is a set of functions serving a common purpose. For example, the LAN profile consists of all functions needed to establish an IP connection

[3]Most of the "phones" that implement the LAN profile are also oviparous, grow wool, and can be milked

which phones are in their proximity, without any application having to run on the users' phones, we had a higher probability of the location data being up to date.

Translating Fixed Nodes

As explained in the previous section, it is desirable that the fixed nodes perform some translation of messages between the phones and the central unit. This spares us from the time-consuming DHCP discovery phase, and the phone does not need to implement any specific profile. It is sufficient if it can send a simple message to the fixed node via Bluetooth. The fixed node can then already verify the message's format, process parts of it, and forward the request to the central unit. The phone does not need any information about the central unit: It communicates only with a fixed node.

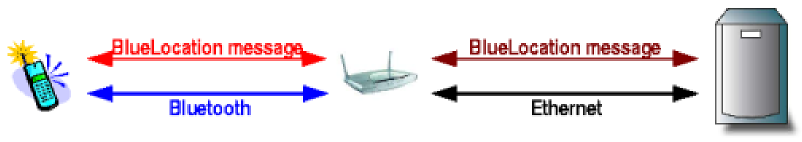


Figure 4.3: Fixed nodes translate all messages

This design allows to change the backend of the system without having to change anything about the phones' application. It requires more powerful, and thus also more expensive fixed nodes as the simple forwarding design.

4.2.2 Hardware Alternatives

Phones

BlueLocation does not have any influence on the kind of phones its users possess. The only restriction that can be imposed on them is that they must be modern enough to support both Bluetooth and third-party Java programs.

Fixed Nodes

The previous section discusses two possible alternatives for the role of the fixed node. We have seen that only the second, which involves computation performed by the fixed node, is feasible in practice. This has an impact on the hardware we choose for the fixed nodes: They have to be able to communicate with the phones via Bluetooth, process phones' requests, and communicate with the central unit, possibly over a large distance. We prefer devices that are easily maintainable: It is not feasible to have fixed nodes whose batteries have to be exchanged after only a few hours of operation. As further requirements, the hardware for the fixed nodes should not be too expensive, their software free and easy to program with. Fixed nodes should be as cheap as possible to encourage people to install BlueLocation, to have great parts of the area covered, and to provide the system for free.

Juggling these requirements has proven not to be easy: Many manufacturers do not equip their Bluetooth devices with an Ethernet or WLAN interface, document their programming interface badly, or refuse to cooperate.

We considered the following devices for use as fixed nodes: Bluetooth access points and routers, single board computers, ETH's own BTnodes [15], PDAs, and laptop or desktop computers.

Bluetooth access points, much like WLAN access points, allow other devices to access the internet. Naturally, they are equipped with interfaces for Bluetooth and Ethernet. Most vendors of such access points advertise their devices as supporting "software upgrades", but remain tight-lipped about the underlying operating system, programming language, and programming interface. For practical reasons, we decided not to get involved with a device we did not know anything about and could not expect any support for.

Figure 4.4: The UsbGear USBG-BLUE-RR Bluetooth access point

Single board computers behave much like fully equipped desktop computers. As their name suggests, all necessary hardware such as graphics or sound cards is integrated into the motherboard. This property also makes them rather small in size. Single board computers are often designed to serve a special purpose such as rendering images or sound. We did not find any device that directly supported Bluetooth, but many that offered a USB interface where we could have plugged in an adapter. Unfortunately, single board computers are optimised towards tasks that are nowhere near our requirements, and so their price is, although justifiable for their designated task, too high for our purposes.

Figure 4.5: Evalue Technology EES-3611 Micro PC

BTnodes are small devices designed for ad hoc Bluetooth networking. They are fully programmable computers, yet they do not provide a WLAN or Ethernet interface. Their hardware could be extended with adapter boards for other systems, but the software for it is still in fledgling stages.

The remaining devices - PDAs, laptops, and desktop computers - are very similar: Their hardware and software supports Bluetooth and Ethernet or WLAN, and they are easy to program. PDAs in particular are very small, rather cheap, and come with all functionality needed.

The following table summarizes our requirements and how the individual devices meet them:

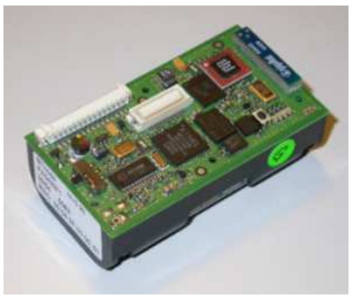


Figure 4.6: ETH's BTnode Rev.3

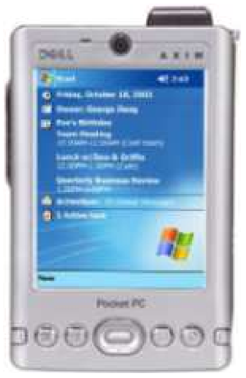


Figure 4.7: Dell's AximX30i PDA can communicate via Bluetooth and WLAN

	uplink	size	price	easy to program	power supply
access point	++	++	+	?	++
single board	++	+	−	+	++
BTnode	-	++	+	+/-	-
desktop	++	−	+/-	++	++
laptop	++	+/-	+/-	++	++
PDA	+	++	+	++	+

We have developped the fixed nodes' software on normal desktop computers, as the application would be portable to PDAs or other future devices providing a similar platform.

Central Unit

The task of the central unit is to communicate with the fixed nodes, to respond to phones' requests, and to store data. This requires that the central unit's hardware has rather big storage, a network interface, and is easily programmable. We chose a desktop computer since it meets all of these requirements, and is not too expensive for purchase and maintenance. Most future system administrators will already have such a computer and will not have to purchase additional hardware.

4.2.3 Software Alternatives

Phones

As we do not have any influence on the hardware of our users' phones, we neither have an influence on their operating system or applications present on the phones. We only request the phones to implement the Java APIs for phone applications (MIDP 2.0 [16]) and Bluetooth (JSR-82 [17]).

Fixed Nodes

The previous section explains that we use desktop computers or PDAs as hardware for fixed nodes. The requirements for fixed node software thus are

- it runs on desktop computers
- it interoperates with Bluetooth, Ethernet or WLAN networking
- it provides support for SSL[4]
- it is free for use and easily programmable

For the latter reasons, we have chosen Linux as the operating system. Linux is - other than other operating systems - portable to many handheld devices, so we can expect the fixed node application to be portable to many devices.

We have chosen Java as the programming language for the fixed nodes because it was the only language which would support SSL in a straightforward way. Other languages that were taken into consideration were C and Python, but had to be abandoned for the reason that the existing SSL library's documentation (OpenSSL [18]) does not fulfil the requirement of being easy to program with, or (in the case of Python) that SSL is not implemented in the language's library.

Central Unit

The central unit consists, as described in section 4.1, of two parts: the database and the controlling logic. The requirements to the operating sytem of the central unit thus are that it runs on the hardware chosen (a desktop computer), supports the database system, and the controlling logic. These requirements are met by most of today's operating systems and also here, Linux was the system of choice for the reasons that its usage is free and it is portable to many hardware configurations.

For the database, we used the PostgreSQL [19] database. We refrained from evaluating many database alternatives against each other as most implementations provide the same functionality and performance.

The controlling logic requires that the implementation language supports SSL, and interfaces with the database. As with the fixed nodes, Java was chosen as it is the only language supporting SSL well enough for our purposes. Java interfaces neatly with the PostgreSQL database.

4.3 Security Issues

4.3.1 Definition of the System and Threat Model

We define three kinds of people who may possibly interact with the BlueLocation system:

[4]This is described in section 4.3

- *System owners* or *administrators* are the people who own a particular BlueLocation system's infrastructure. This could be, for example, the organizing committee of a ball. They are allowed to interact directly with the fixed nodes or the central unit. By "directly" we mean any interaction taken not via messages within the BlueLocation system - for example, a command entered in a console.
- *Users* of the BlueLocation system are people who have the BlueLocation phone application installed on their phone. They may not directly interact with the system, but they may interact indirectly with BlueLocation via the commands the phone application offers them.
- Any other person trying to interact directly or indirectly with the BlueLocation system is considered an *attacker*.

A fully running BlueLocation system keeps track of where and when a Bluetooth enabled device was seen. Nobody but this BlueLocation system's owner has insight into the data gathered and stored about this information, save for the case when someone known to be a friend of a person requests to know where that person is. The system's owner has access to all data stored. The data stored can be altered, updated, or deleted only by the following two actions: An explicit interaction of the system administrator with the central unit or an action taken by a user that is legal as described below.

BlueLocation provides users with information about another person's current location if that person declares the requesting user to be their friend. A user can tell BlueLocation who her friends are. BlueLocation will consider everybody else a foe, and it will not deliver any location information about a distinct user to a foe. Only the administrator, the user declaring someone to be his friend and the user being declared as a friend know about this relationship. For a user, it is not possible to find out about another distinct user's friends. Nobody but the user declaring someone as a friend can change the status of that person back to foe or vice versa. It is not possible to pretend to have another person's identity, to inject false data, or to take any other action described as illegal above without a considerable effort.

In particular, knowledge about the underlying protocols and implementation, tapping the communication between fixed nodes and the central unit, mocking up and replaying any message are considered to be common threats, and BlueLocation defends against those. BlueLocation does not defend against theft or failure of any hardware, failure of the communication channels used, sophisticated mock-up of a person's identity, such as faking the Bluetooth ID of the device used, any unauthorised access to the central unit or a fixed node that is due to a lack of authorising mechanisms at the central unit or fixed node itself, and tapping of the Bluetooth communication.

By tapping the Bluetooth connection, the attacker cannot, by the definition of the BlueLocation system, fool the system into believing her to be another person, alter any data, or gather any data about a distinct user.

The implementation of the BlueLocation system ensures the functionality described above. The precautions taken to prevent actions described as illegal above are described in the next section.

4.3.2 Precautions

Authentication

BlueLocation must ensure the identity of users and system owners. Authentication of system owners is left to the authentication mechnisms of the fixed nodes and the central unit. BlueLocation has no possibility to distinguish an intruder interacting directly with a fixed node or the central unit from someone authorised to do so.

BlueLocation users are identified by their mobile phone's Bluetooth ID. This serves two purposes: For one, the Bluetooth ID of every device is unique. It is non-trivial and a considerable effort to send a Bluetooth message with a mocked-up Bluetooth ID. Any fixed node communicating with a phone retrieves the Bluetooth ID of that phone from the low level networking layers, and appends this information to the message sent to the central unit. This ensures that assuming a false identity by mocking up a Bluetooth ID is non-trivial.

Secure Communication

Bluetooth communication is, as described in the previous section, not protected. The communication between the central unit and the fixed nodes is encrypted with SSL [14], which ensures both authentic and secure (non-tappable) communication. This communication channel, as opposed to the Bluetooth channel, must be protected: Other than with Bluetooth, immediate physical proximity of the attacker is not necessary. Messages injected into this channel can per se not be checked for authenticity, not even with the help of lower level networking information. The possibility of an educated attacker mocking up such information is considerably higher as the number of possible attackers is much higher without the exigence of physical proximity.

Since an attacker cannot directly interact with a fixed node or the central unit, encoding messages with SSL is enough to ensure that it is not possible to tap, inject, change, or replay messages.

Nonproliferation of Stored Data

We can assume that all users interacting with the system are authenticated as described in the previous section. We do not have to take any precautions against users interacting directly with a fixed node or the central unit as we can assume them to be administrators.

The only case left is an authenticated user sending a BlueLocation message. When the central unit receives such a request, it checks whether the user has the appropriate rights, and only answers the request if that is the case.

Nonaltering and Nondeleting of Stored Data

Following the above reasoning, we can assume any user interacting with the system to be authenticated. Users interacting directly with a fixed node or the central unit are considered to be administrators, and are allowed to alter and delete any data. Users interacting indirectly with the BlueLocation system can only send BlueLocation messages to a fixed node or to the central unit. All messages are authenticated. Thus, any alternation or deleting of data stored is authenticated as well and thus allowed.

Chapter 5

Implementation

5.1 Phones' Design

The phones' task is to offer the user a list of actions she can take, to send a request corresponding to such an action to a nearby BlueLocation fixed node, to receive the answer from that node, and to display it to the user. The phone does not have to periodically send information about its location to the central unit: The fixed nodes can scan for Bluetooth enabled devices in their proximity and send this information to the central unit. This allows locating phones even when they have the BlueLocation application turned off.

The BlueLocation phone application does not keep any state about its user. All information is assumed to be stored at the central unit. This allows the user to install BlueLocation on his mobile phone once and then use it whereever a BlueLocation system is running. Another advantage of this setting is that there are less requirements to the software support of the phones: They need not provide support for storing data.

At startup of the application, the phone must retrieve data about the current environment it is running in. This consists of the user's user id (a human-friendly number that uniquely identifies this person in the current BlueLocation system) and two lists: One list contains all people that allow this user to search for them (these people are referred to as *granters* or *contacts*), and a second list that contains all users that are allowed to search for this user (referred to as *grantees* or *peers*).

When the application has started up and gathered this initial information, the user can take one of the following actions:

- look at the list of people whom he has granted the right to search for him, giving a further person this right, or revoke this right from one of them
- look at the list of people who have granted this user the right to search for her, ask BlueLocation to look up where such a person was last seen
- shut down the application

When BlueLocation is developped further, it will also allow the user to switch mobile phones and to give each peer and each contact a user readable name (much like people do with persons whose contact information they store on their mobile phones).

5.2 Fixed Nodes' Design

Fixed nodes perform two tasks:

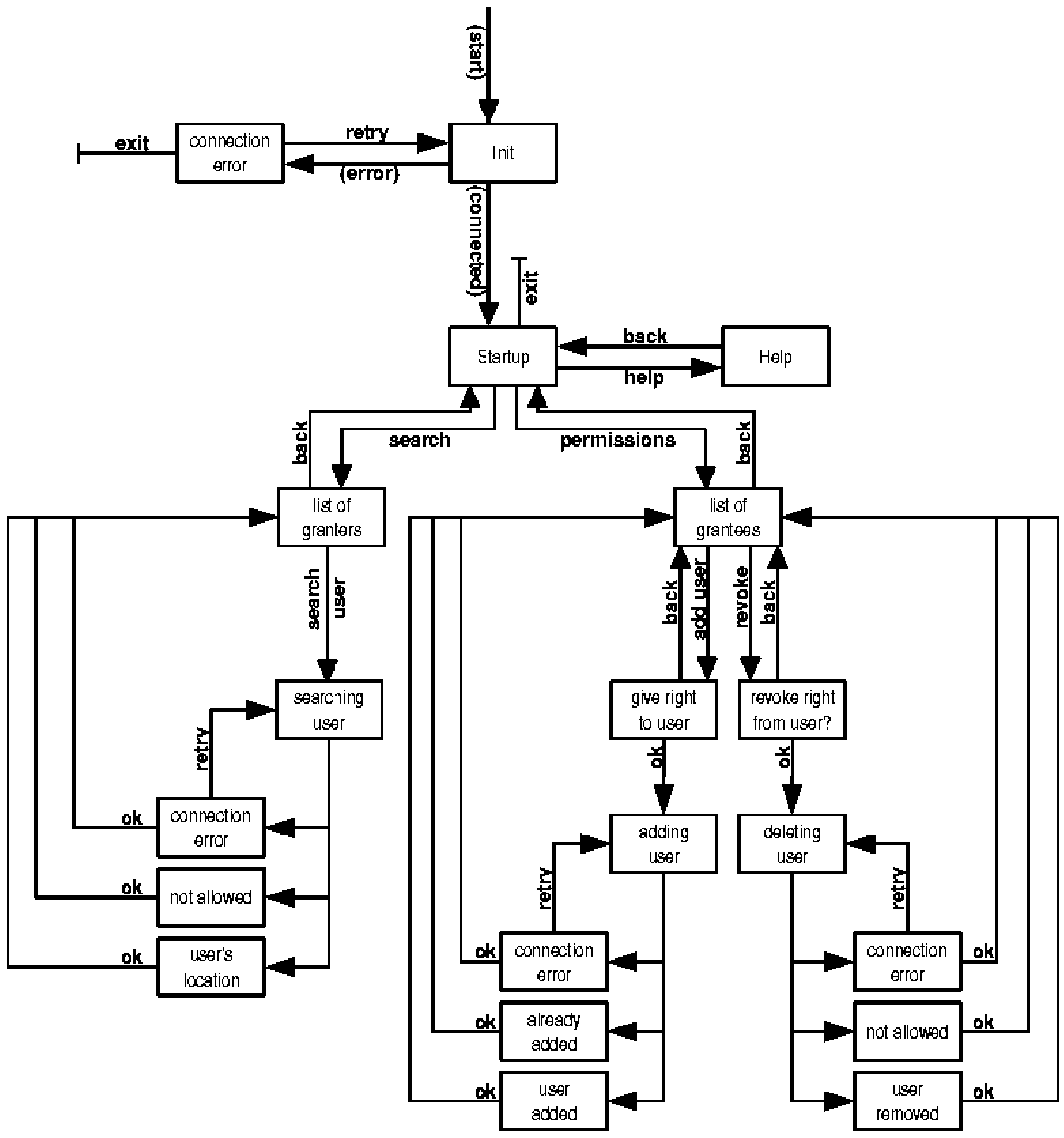


Figure 5.1: phone application state diagram

- periodically scan their neighbourhood for devices that have Bluetooth turned on and report the seen devices to the central unit
- accept a phone's request, add the requesting phone's Bluetooth ID to the request (as described in section 4.3), forward the request to the central unit, and return the central unit's response to the requesting phone

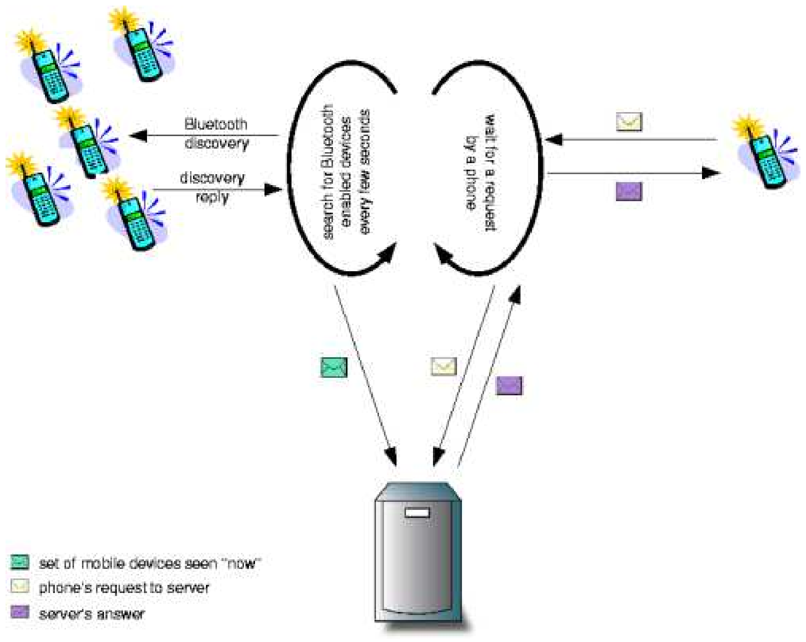

Bluetooth discovery
discovery reply
search for Bluetooth enabled devices every few seconds
wait for a request by a phone
set of mobile devices seen "now"
phone's request to server
server's answer

Figure 5.2: The fixed node consists of two independent threads

Each of these actions is implemented as an independent thread. One thread constantly listens to incoming phone requests, and when it receives a request, it splits a worker thread that handles it to keep the original listening thread from blocking. The worker thread then sends the request to the central unit, waits for an answer, and sends it back to the requesting phone.

A second thread waits for a given amount of time, performs a Bluetooh inquiry, constructs a message for the central unit reporting all devices it has found, sends it to the central unit, and goes back to wait.

5.3 Central Unit Design

5.3.1 Controlling Logic

The central unit is implemented using the factory pattern: A main thread waits for a fixed node to contact it. When it receives a message from a fixed node, it determines which kind of message it is, and splits an appropriate worker thread. This keeps the main thread from blocking on a call. The worker thread then checks whether the message has the correct format and possibly verifies that a user requesting an information or a change in the database has the correct permissions to do so. If it finds everything to be correct, it performs the requested operations, and returns an appropriate answer to the fixed node. If it finds that the message contains an error or requests an illegal action, it returns an error message.

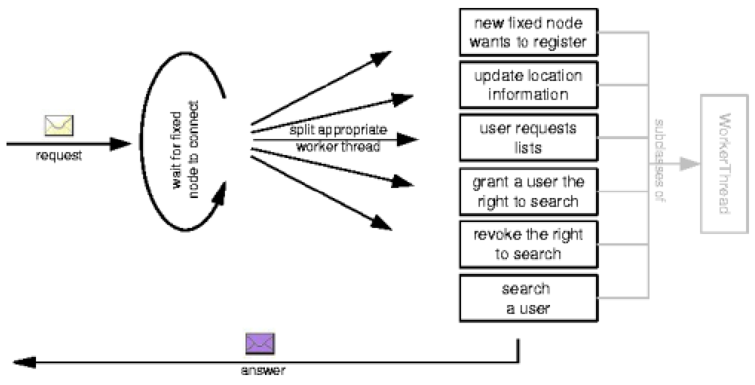

request
wait for fixed node to connect
split appropriate worker thread
new fixed node wants to register
update location information
user requests lists
grant a user the right to search
revoke the right to search
search a user
subclasses of
WorkerThread
answer

Figure 5.3: The factory method splits an appropriate worker thread for each message

5.3.2 Database Design

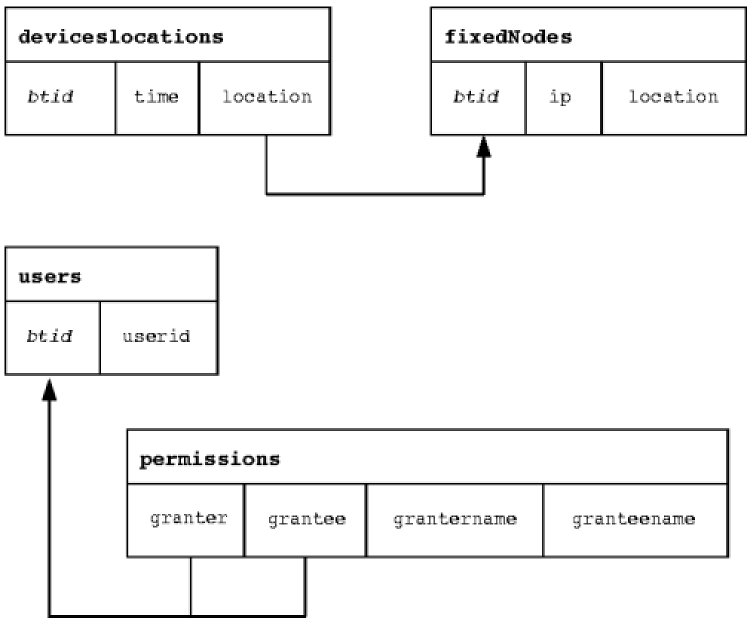


Figure 5.4: BlueLocation database schema

The database must store all data BlueLocation collects. This data falls in two categories: Location information, and BlueLocation user information. Location information contains data about when a certain Bluetooth enabled device was seen at which fixed node.

The user information part keeps track of which user has allowed which another users to search for him. This part is completely independent of the location information and can be left away if BlueLocation is used simply to gather location information on Bluetooth enabled devices.

5.4 Protocols

5.4.1 Phone Requests

Semantically, a phone always interacts with the central unit: Even though the phone connects to a nearby fixed node, the fixed node only inserts the Bluetooth ID of that phone into the request, and forwards it to the central unit. From the central unit's answer, the fixed node strips the Bluetooth ID, and sends the answer to the phone. Therefore, the messages exchanged between the phone and the fixed node are the same as the messages forwarded to the central unit, except for the requesting phone's Bluetooth ID.

A phone can send the following requests to the central unit:

- request initial information (`ListRequest`). This message is sent at startup of the application and can be triggered to update the information during the time the application runs. Initial information contains this user's user ID and the two lists for peers and contacts.
- grant someone the right to search for this user (`AddUser`). This results in an update of the database. The central unit only informs the phone of the success of this action: The phone then still has to re-retrieve the peer and contacts lists for updates. The lists are not pushed to the phones to keep the number of messages sent low as possible.
- revoke the right to search for this user from someone (`DelUser`). Similarily, this request also results in an update of the database and information about the success of the action. The lists are not pushed to the phone.
- ask where someone is (`SearchUser`). If the user requesting the information is a peer of the person he is looking for, the central unit responds with the location that user was last seen at. It can however be the case that due to the lazy updating of peer and contacts lists, the user still has a person in his phone's contact list who in the mean time revoked the right to search for her. In this case, the central unit returns an error message.

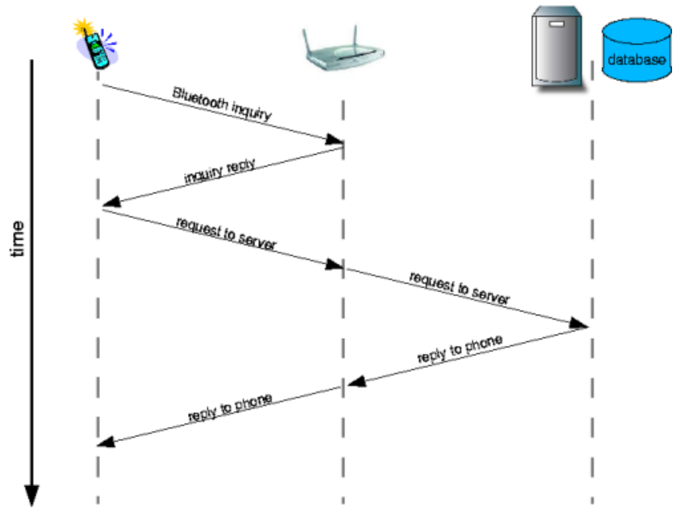


Figure 5.5: After the Bluetooth inquiry, the phone communicates with the central unit via a fixed node

`PhoneMessage:`	`ListRequest | AddUser | DelUser | SearchUser`
`ListRequest:`	`’listrequest?’ BTID ’?’ GranterList ’?’ SequenceNumber`
`AddUser:`	`’adduser?’ BTID ’?’ GranteeUID ’?’ GranteeName ’?’ SequenceNumber`
`DelUser:`	`’deluser?’ BTID ’?’ GranteeBTID ’?’ SequenceNumber`
`SearchUser:`	`’searchuser?’ BTID ’?’ GranterBTID ’?’ SequenceNumber`
`BTID:`	*the Bluetooth ID of the requesting phone*
`GranterList:`	`Granter | GranterList ’’ Granter`
`Granter:`	`GranterBTID ’!’ GranterName | ’’`
`SequenceNumber:`	*the sequence number of this message*
`GranterBTID:`	*the Bluetooth ID of the granter*
`GranteeBTID:`	*the Bluetooth ID of the grantee*
`GranteeUID:`	*the user ID of the grantee*
`GranterName:`	*the grantee-given name of this granter*
`GranteeName:`	*the granter-given name of this grantee*
`AnswerToPhone:`	`ListAnswer | OtherAnswer`
`ListAnswer:`	`BTID ’?’ UID ’?’ ContactList ’?’ PeerList ’?’ SequenceNumber`
`OtherAnswer:`	`( ’good:’ | ’error:’ ) Answer ’?’ SequenceNumber`
`ContactList:`	`Contact | ContactList ’!’ Contact`
`Contact:`	`ContactBTID ’’ ContactUID ’’ ContactName | ’’`
`PeerList:`	`Peer | PeerList ’!’ Peer`
`Peer:`	`PeerBTID ’’ PeerUID ’’ PeerName | ’’`
`UID:`	*the user ID of the requesting phone*
`ContactBTID:`	*the Bluetooth ID of this contact*
`PeerBTID:`	*the Blueotooth ID of this peer*
`ContactUID:`	*the user ID of this contact*
`PeerUID:`	*the user ID of this peer*
`ContactName:`	*the grantee-given name of this contact*
`PeerName:`	*the granter-given name of this peer*
`Answer:`	*A human-readable answer sentence*

Table 5.1: messages exchanged between phones and the central unit

5.4.2 Fixed Node / Central Unit Protocol

Besides for the forwarding of the phones' requests, fixed nodes also interact with the central unit to report Bluetooth devices in their proximity: They periodically send a list of devices they have seen to the central unit. The fixed nodes do not wait for an answer by the central unit because the underlying TCP protocol guarantees that a message sent eventually arrives its destination, given the destination is reachable. So if the central unit is not reachable, it could also not send an error message back to the fixed node. In a more sophisticated implementation of the BlueLocation system, the fixed node could buffer a certain amount of data and send that forward to the central unit when it was reachable again. We did not implement this as we assumed that once the central unit was unreachable, it would remain so for a considerably long time, such that the storage of the fixed node would not be enough to buffer all data gathered in the meantime. Fixed nodes are expected to be small devices, and we did not want to require them to have a big storage to support buffering of data.

Therefore, there is only one message sent from the fixed node to the central unit. It takes the following form:

```
'locationupdate?' { BTID '!' }
```

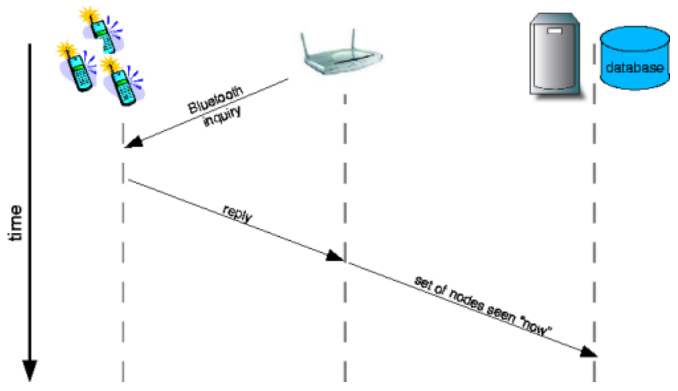


Figure 5.6: after it has discovered nearby Bluetooth devices, the fixed node reports them to the central unit

Chapter 6

Outlook and Conclusion

6.1 Conclusion

In this thesis, we have described and implemented a system that collects location information about mobile users over time. It does so by using the users' mobile phones' Bluetooth radio as an indicator of the presence of a user. Location information is gathered by so-called fixed nodes which periodically scan their proximity for Bluetooth enabled devices and report the seen devices to a central unit. It is assumed that the location of the fixed nodes is known, and by the relatively short range of the Bluetooth radio, it can be assumed that any device reported by a fixed node is at the same location as that fixed node.

In addition, this thesis proposes an application for the data gathered: It describes an application for mobile phones that, by interacting with the system described above, can retrieve data abou the current location of other users.

6.2 Optimisations to the Current System

The BlueLocation system as proposed and implemented in this thesis may not be optimal in many ways. Further theses will evaluate, adapt, and extend the system. We suggest investigating the following optimisations:

- The database can be sped up by adding redundant tables for quick lookups, for example the last location a person was seen at.
- The protocol implemented in the current BlueLocation system is human-readable and contains redundancy. Future adaptions of the BlueLocation system can take load from the Bluetooth connection between a phone and a fixed node by removing redundancy of the information exchanged from the messages and by applying compression. It is also possible that the amount of information exchanged can be reduced by introducing statefulness at the phones. Simultaneously, feasibility and efficiency of encrypting the Bluetooth connection could be evaluated.
- The current phone implementation is very minimal. An implementation targeted for use in a real environment will need more bells and whistles, such as letting a user upload an avatar image of himself, allow a user to switch phones, sending other users messages via BlueLocation, and informing the user when one of his friends enters the Bluetooth range of the phone or when someone is looking for her.

6.3 Future Extensions

This thesis is the start of more theses in the area of mobile user locating using Bluetooth. BlueLocation can be extended by applications computing and showing statistics, real-time visualisations, or simulations based on the data gathered. On the user-friendly side, applications using the location information can adapt their behavior based on where a user currently is.

Appendix A

Full Task Description

Institut für Technische Informatik und Kommunikationsnetze

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo

Communications Systems Research Group

Summer Semester 2005

Semester Thesis

for

Katrin Bretscher (D-INFK), Daniel Hottinger (D-INFK)

Advisors:	Vincent Lenders and Matthias Bossardt
Supervisors:	Dr. Martin May and Prof. Dr. Bernhard Plattner

Issue Date:	29th March 2005
Submission Date:	1st July 2005

BlueLocator - A Locating Infrastructure for Bluetooth Enabled Mobile Phones

1 Summary

Most of today's mobile phones feature Bluetooth for short range communication. In addition of using Bluetooth for communication purposes, it is also imaginable to use Bluetooth for locating purposes. The position of a mobile user can easily be determined by placing fixed Bluetooth nodes at pre-determined positions and checking connectivity of the user with the fixed nodes, since connectivity information directly indicates the proximity of a user to a fixed node. Many applications require postition information of users. For example, a location service can be provided to users at large events to find friends using their mobile phones. Another straighforward usage of a locating infrastructure is to establish mobility patterns by collecting position information of mobile users at various places over time. The latter usage is specially interesting to study the behavior of mobile users which is of high interest in the networking research community.

The goal of this thesis is develop a locating infrastructure for indoor purposes. Such an infrastructure consists of a certain number of fixed Bluetooth nodes to determine connectivity of mobile users, and a central instance that collects and keeps track of position information. The task of the students is to design the whole infrastructure and implement the required system instances. The system shoud be flexible enough to support both aformentioned applications scenarios. A prototype of the locating infrastructure should be build at the lab to demonstrate its applicability.

This thesis is part of the Blue-* project [1].

2 Assignment

2.1 Tasks

- Identify related work and look for similar projects.
- Familiarize yourself with Bluetooth.
- Specify the hardware requirements of a suitable fixed Bluetooth node.
- Design the locating infrastructure
- Design the required communication protocols.
- Develop a localization algorithm.
- Implement the whole locating infrastructure.
- Setup a prototype infrastructure at the lab.

2.2 Deliverables

- At the end of the second week, a detailed time schedule of the semester thesis must be given and discussed with the advisor.
- At half time of the semester thesis, a short discussion of 15 minutes with the professor and the advisor will take place. The student has to talk about the major aspects of the ongoing work. At this point, the student should already have a preliminary version of the written report, including a table of contents. This preliminary version should be brought along to the short discussion.
- At the end of the semester thesis, a presentation of 20 minutes must be given during the TIK or the communication systems group meeting. It should give an overview as well as the most important details of the work and the demonstrator should be presented at this time.
- The final report may be written in English or German. It must contain a summary written in both English and German, the assignment and the time schedule. Its structure should include an introduction, an analysis of related work, and a complete documentation of all used software tools. Three copies of the final report must be delivered to TIK.

References

[1] *The BlueStar Project*, http://www.csg.ethz.ch/research/running/Blue_star, October 2004.

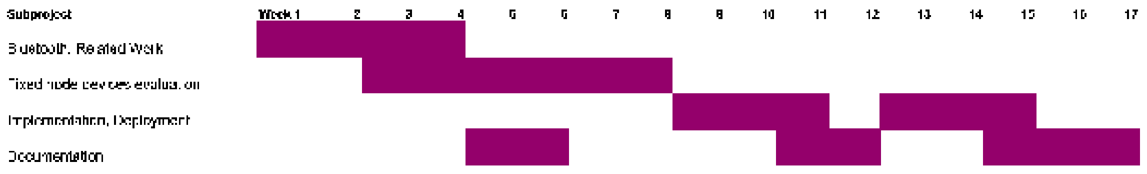
31th March 2005

Prof. B. Plattner

2

Appendix B

Project Timeline



Subproject	Week 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Bluetooth, Related Work	■	■	■	■													
Fixed node devices evaluation			■	■	■	■	■	■									
Implementation, Deployment									■	■	■		■	■	■		
Documentation					■	■					■	■			■	■	■

Appendix C

Deliverables

The CD accompanying this document contains all code, scripts, manuals for the BlueLocation system. It has the following structure:

- **phone application**
  - application design
  - core implementation
- **fixed nodes**
  - application design
  - full implementation
  - testing routines
  - install scripts
  - installation guide
- **central unit**
  - application design
  - database design
  - full implementation in
  - testing routines
  - install scripts
  - installation guide
- **documentation**
  - this document
  - slides used for the presentation

Bibliography

[1] The Global Positioning System, run by the U.S. Department of Defense
`http://www.defenselink.mil`

[2] The Bluetooth radio technology
`http://www.bluetooth.org`

[3] The Blue* project
`http://www.csg.ethz.ch/research/projects/Blue_star/`

[4] Ganymed Stanek: *Bluetella: A Java Application For New Mobile Phones*, Semester Thesis TIK-SA-2003.19, ETH Zurich, July 2003.
`ftp://www.tik.ee.ethz.ch/pub/students/2003-So/SA-2003-18.pdf`

[5] Andreas Weibel, Lukas Winterhalter: *Bluetella: File Sharing for Bluetooth Enabled Mobile Phones*, Semester Thesis TIK-SA-2005.3, ETH Zurich, February 2005.
`ftp://www.tik.ee.ethz.ch/pub/students/2004-2005-Wi/SA-2005-03.pdf`

[6] Sandro Schifferle, Christian Braun: *BlueDating: A Dating Application for Bluetooth Enabled Mobile Phones*, Semester Thesis TIK-SA-2005.8, ETH Zurich, February 2005.
`ftp://www.tik.ee.ethz.ch/pub/students/2004-2005-Wi/SA-2005-08.pdf`

[7] Nicole Hatt: *BlueStar: An Application Framework for Wireless Ad-Hoc Networks*, Master Thesis, ETH Zurich 2005. (Not published at the time of writing)

[8] buZZone
`http://www.buzzone.net`

[9] mobiLuck
`http://www.mobiluck.com`

[10] proxidating
`http://www.proxidating.com`

[11] nTag
`http://www.ntag.com`

[12] spotMe
`http://www.spotme.ch`

[13] Speck
`http://speck.randomfoo.net`

[14] Secure Socket Layer `http://en.wikipedia.org/wiki/Secure_Sockets_Layer`
`http://en.wikipedia.org/wiki/Netscape_Communications_Corporation`

[15] BTnodes - A Distributed Environment for Prototyping AdHoc Networks
http://www.btnode.ethz.ch

[16] Forum Nokia: MIDP 2.0: Introduction
http://www.forum.nokia.com/ndsCookieBuilder?fileParamID=6516

[17] Java APIs for Bluetooth Wireless Technology (JSR-82)
http://jcp.org/aboutJava/communityprocess/final/jsr082/index.html

[18] OpenSSL: Open Source Toolkit for SSL/TLS
http://www.openssl.org

[19] PostgreSQL open source database
http://www.postgresql.org