

Reto Strahl

Build Your Own World

Student Thesis SA-2001.28

Summer Term 2001

Tutor: Jan Beutel

Supervisor:

Prof. Dr. Lothar Thiele

Build Your Own World

Reto Strahl

17th July 2001

Abstract

For future applications using distributed radio networks, positioning data will be essential. The current GPS system does not work indoors. Therefore we have to find other systems to estimate the position of the radio transceivers. The position data can be estimated knowing the position and distance of other transceivers in the close environment. To build up test environments we focus on Bluetooth networks. To estimate the distances between two Bluetooth devices signal strength measurements on connections are used. In this work a simulation tool is created to investigate different positioning algorithms. The simulation has a modular design so it will be possible to distribute parts of the simulation to Bluetooth nodes.

Acknowledgment

This documentation is the result of a term project at the Department of Electrical Engineering at the ETHZ. During the last four years at the Institute, two term projects needed to be done. This is my first project. I'm sure the experience from this project will help me master the second term project.

During the work on this project I have learned more about object oriented design in C++. Further, I tried to profit from the benefits of the STL (standard template library). The more I know about C++ the more I'm getting lost in the various implementation possibilities this language is offering. For instance, I never thought I would use a pointer to a member function..

I also would like to thank a few people helping me on the term project. First of all Jan Beutel. He has been guiding and inspiring me. It is amazing to work with someone having a lot of good ideas in every meeting. It was to bad that I was not able to implement more of them. The interesting work with the algorithms themselves will be left for him or an other student. Secondly, Sabine Bernhard my girlfriend . She had to suffer each time I turned my computer on late in the evening. Thirdly my parents and my boss Mr. Stocker for providing me with the little money one needs to survive in Zurich. I hope I didn't forget anyone and if I have I feel extremely sorry. Just append your name with a pen on your personal copy of this document.

Contents

<i>1: Introduction</i>	<i>1</i>
<i>2: Background</i>	<i>3</i>
2.1 Triangulation with Range Measurements	3
2.1.1 Triangulation for Absolute Positions	3
2.2 Ad-hoc Wireless Networks	4
2.2.1 Bluetooth	5
<i>3: Simulator Concept</i>	<i>7</i>
3.1 Development Environment	7
3.2 Basic Concept	8
3.3 Positioning Process	9
<i>4: Implementation</i>	<i>11</i>
4.1 Compilation	11
4.1.1 Visual C++	11
4.1.2 Linux	11
4.2 Class Implementations	12
4.2.1 Controller Class	12
4.2.2 SimulatePositions Class	12
4.3 BluetoothNode Class	13
4.4 PositionNode Class	14
<i>5: Summary & Outlook</i>	<i>15</i>

1

Introduction

PicoRadio [1] networks consist of independent radio transceivers transmitting data over short distance. These radio transceivers are called PicoNodes. The PicoNodes operate with low power consumption and allow flexible communication. Ad-hoc networks are characterized by the ability of the nodes to enter and leave a network independent.

Many future applications will use the attractive possibilities of ad-hoc PicoRadio networks. Sensor networks controlling the environment, smart rooms and inventory systems are only a few examples. A lot of applications will rely on position data. Sensor data for instance is almost useless without time and position data.

Bluetooth [2] is a new communication standard. It allows to setup communication channels between mobile radio transceivers. Bluetooth devices can be used to build up ad-hoc and PicoRadio networks. Ad-hoc network means that the nodes can enter and leave the network independent. The nodes can inquire the environment to look for potential communication partners. Once a node has detected another node it can setup a communication channel and start communicating. Bluetooth transceivers will be used in mobile and fixed devices like PDA's, phones, headphones, PC's or printers.

The goal of this term work is to setup a simulator to examine different algorithms. The simulation should take care of a Bluetooth environment. The simulator is written in C++ using object oriented programming. The simulator can be compiled for UNIX and Linux as a simple console application. Further a simple GUI for Windows was generated using MFC.

2

Background

This chapter gives an introduction to the technical background of the simulation. The first section explains the mathematics needed to estimate the position. The second section will give a brief overview of ad-hoc networks and the Bluetooth standard.

2.1 Triangulation with Range Measurements

Triangulation methods allow a remote position to be determined using range and angle data. Range data for radionavigation can be measured using TDOA (time difference of arrival), TOA (time of arrival) or RSSI (Received signal strength). Triangulation [1] can be applied to compute absolute positions or just position relative to others. In this document we will focus on triangulation to compute absolute positions.

2.1.1 Triangulation for Absolute Positions

Absolute positions can be calculated using range data to known positions.

To determine the exact location of a radio navigation receiver multiple range measurements and absolute positions must be known. In 3D at least four measurement tuples (range and corresponding absolute position) must be given. For 2D at least three tuples must be known.

Assumed that we want to compute the 2D coordinates U_x, U_y using n given positions and range measurements the navigation equations would be:

$$\begin{bmatrix} (X_1 - U_x)^2 + (Y_1 - U_y)^2 \\ (X_2 - U_x)^2 + (Y_2 - U_y)^2 \\ \vdots \\ (X_n - U_x)^2 + (Y_n - U_y)^2 \end{bmatrix} = \begin{bmatrix} R_1^2 \\ R_2^2 \\ \vdots \\ R_n^2 \end{bmatrix} \quad (2.1)$$

X_i, Y_i are the position coordinates of the transmitter i . R_i is the range measured for transmitter i . For more than three range measurements the system is overdetermined. Minimum mean square error methods (MMSE) can be used to solve the overdetermined system.

The system can be linearized by subtracting the last line from the other lines. The new linear system has the form:

$$2\mathbf{X}\mathbf{u} = \mathbf{r} \quad (2.2)$$

Where the symbols \mathbf{X} , \mathbf{u} and \mathbf{r} have the form:

$$\mathbf{X} = \begin{bmatrix} (X_n - X_1) & (Y_n - Y_1) \\ (X_n - X_2) & (Y_n - Y_2) \\ \vdots & \vdots \\ (X_n - X_{n-1}) & (Y_n - Y_{n-1}) \end{bmatrix}$$

$$\mathbf{u} = \begin{bmatrix} U_x \\ U_y \end{bmatrix}$$

$$\mathbf{r} = \begin{bmatrix} R_1^2 - R_n^2 - X_1^2 - Y_1^2 + X_n^2 + Y_n^2 \\ R_2^2 - R_n^2 - X_2^2 - Y_2^2 + X_n^2 + Y_n^2 \\ \vdots \\ R_{n-1}^2 - R_n^2 - X_{n-1}^2 - Y_{n-1}^2 + X_n^2 + Y_n^2 \end{bmatrix}$$

The linearized system can be solved using a MMSE for overdetermined systems such as the QR decomposition algorithm.

2.2 Ad-hoc Wireless Networks

Ad-hoc wireless Networks are non static networks. They are based on wireless Transmission. Their topology may change unpredictable.

The networks don't have a central coordination. There is no central entity managing the network configuration or monitoring the network topology. The network nodes are free to enter and leave the network. The nodes can also move within the network. This means that they can change their location as well as the entry points to the network.

The network nodes have just a small knowledge about the network. A node just knows about the space he his transmitting to. Without transmitting further information a node just knows that there are other nodes around but he does not know where they are located. To gain bigger knowledge about the topology or his own position in the network interaction between the network nodes is needed.

2.2.1 Bluetooth

Bluetooth [2] is a new communication standard. It allows to transmit data and audio over short distances. Bluetooth nodes can form pico nets containing up to 8 nodes. Several pico nets can form bigger scatter nets. Bluetooth nodes can communicate on an ad-hoc basis. They need no user interaction to communicate with other nodes.

Some future applications for Bluetooth might be:

- Replacing cables (printer, PC, personal digital assistant, mouse, keyboard, phones, headsets)
- Automatic synchronization between PDA's, PC, phones
- Remote controls
- Navigation applications
- Sensor and actor networks
- Wireless headsets, microphones, displays

Bluetooth operates in the Industrial-Scientific-Medical (ISM) band at 2.4 GHz. This band is worldwide available. The ISM band is also used by other devices such as cordless phones or microwave ovens. To reduce the interference Bluetooth is using frequency hopping.

Technical data:

- 2.4 GHz band, bandwidth 79 MHz
- Frequency hopping, 1600 changes per second
- 10-100 m transmission range
- Maximal bandwidth 1 Mbit per second
- Radiation power 1 mW

The Bluetooth protocol stack (see figure 2-1) defines several layers:

The Radio Frequency (RF) layer sends and receives modulated bitstreams. It defines the properties of the receiver and transmitters.

Baseband (BB) defines timing, framing and packet flow control on the link. Baseband provides transmission channels for both voice as well as data communication. Link Manager assumes responsibility of managing connections, power management and enforcing fairness among slaves. It further handles link setup, security and device discovery. The L2CAP (Logical Link Control and Adaption Protocol) layer handles multiplexing of higher level protocols, segmentation and reassembly. It provides services to upper layer protocols by transmitting data packets over L2CAP channels. Upon establishment of a connection over a channel, L2CAP negotiates several parameters such as MTU, QOS, time-outs etc.

On top of L2CAP several communication protocols are located. RFCOMM (serial interface emulation) and TCP/IP are just two of them.

Finally the application layer enables applications to use Bluetooth communication. The applications are using the services of the lower protocol layers.

The key functionalities of a Bluetooth device for positioning are:

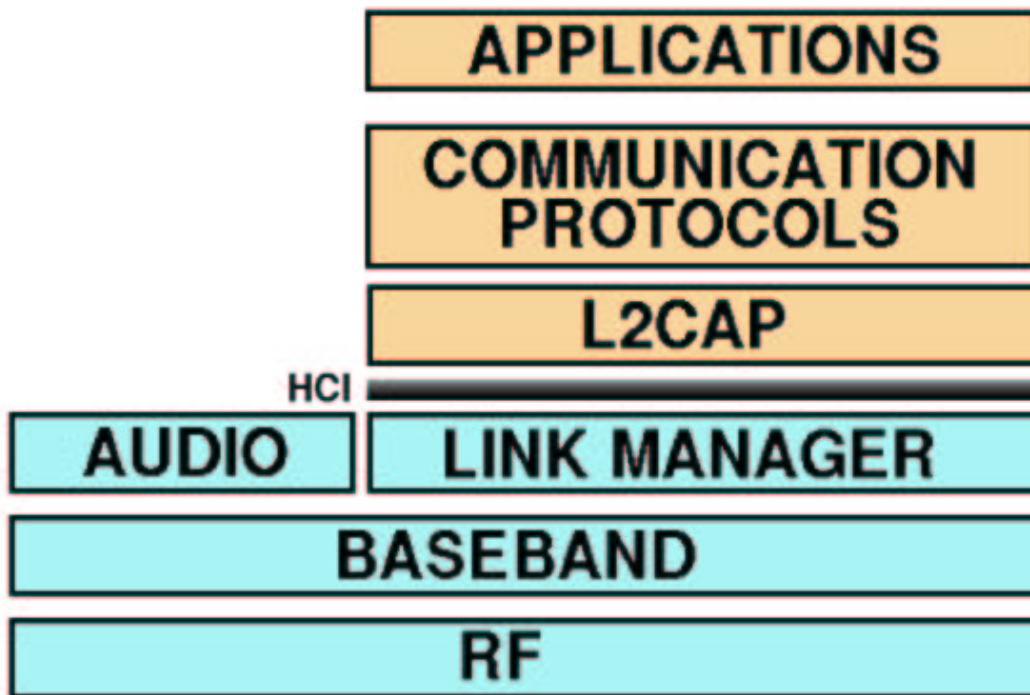


Figure 2-1
The Bluetooth Protocol Stack

- The Bluetooth device can search the environment for other Bluetooth devices. This service is called an inquiry.
- The device can open a connection to an other device in the neighborhood and ask for specific data.
- The device can direct instructions to neighbor devices.
- A RSSI measurement can be issued on a connection.

3

Simulator Concept

An Ad-hoc network doesn't have a central coordination. Therefore, it makes little sense to introduce a central entity to calculate the position data. Every node should try to estimate its position using the data it can directly grab from its neighborhood. In other words, every Bluetooth device estimates and stores its position data by itself.

The main design goal was to write a modular simulation. Different algorithms for the positioning should be tested in the future. It is essential that just the implementation of the algorithms need to be changed and not the hole simulation.

Further different simulation data should be used. For functional testing issues, random generated data is used. Also, data from Bluetooth devices should be interfaced in the future. To switch between these two setups, just one module should be replaced. The rest of the simulation should not change.

Object oriented design is a good approach to achieve the desired modularization.

Another goal was to produce reusable code. The positioning application might be distributed to various Bluetooth nodes. The algorithms tested with the simulation should be portable and reusable on the environment of the Bluetooth nodes.

To develop reusable code object oriented design is a valuable tool. Further it has to be considered that the code must be linkable with the environment of the Bluetooth nodes.

The first section gives an overview of the development environment and the libraries used. The second section introduces the basic concept. The last section traces the positioning process.

3.1 Development Environment

The Bluetooth environment consist of Ericsson Bluetooth Modules and an open source Bluetooth Stack from Axis [4].

The simulator is written in C++. C++ was chosen to meet the requirements described in the introduction above. The Axis Bluetooth Stack is written in C. As C++

is derived from C it is possible to link C code with C++.

The simulation was developed under the Linux [6] operating system. Later a GUI was added which is running under Microsoft Windows. The GUI was programmed using Microsoft Visual C++.

To calculate the triangulation the newmath09 [5] C++ library is used. The library is an open source math library. After some minor changes the code compiled in the Linux and the Windows environment.

3.2 Basic Concept

The simulation has three logical parts as shown in figure 3-1. It consists of the Controller, the Simulation Setup and the Bluetooth Nodes.

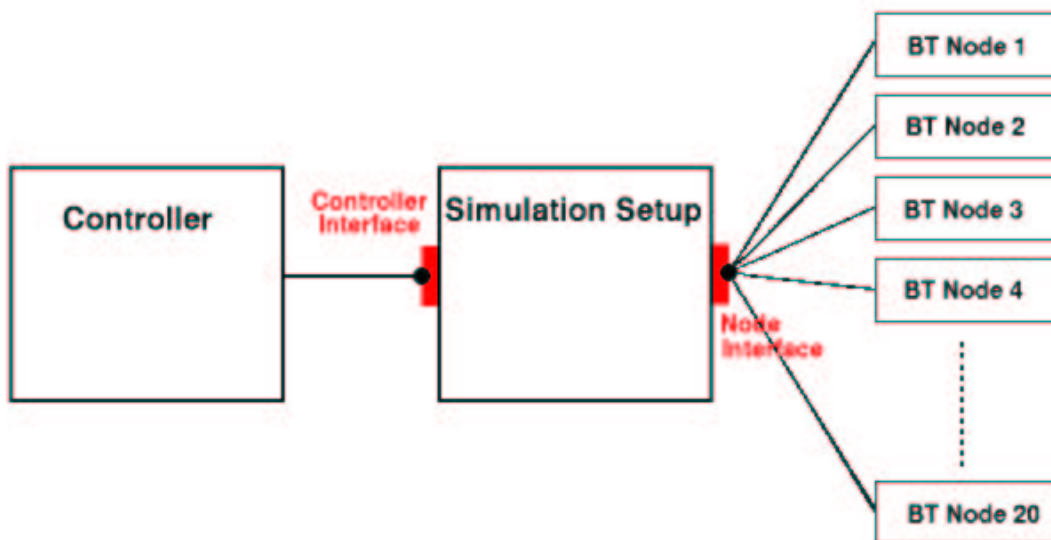


Figure 3-1
The Structure of the simulation is divided in three logical Parts: the Controller, the Simulation Setup and the Bluetooth Nodes.

The Bluetooth Nodes model real Bluetooth devices. They contain the basic functionality of a Bluetooth Node as well as the actual algorithm to estimate their own position.

The Controller is the human interface. It implements a set of commands which will be executed by the simulation.

The Simulation Setup is the core of the simulation. It holds the data required to run the simulation. It also manages the Bluetooth Nodes. The Simulation Setup routes the commands coming from the Controller to the Bluetooth Nodes. The Bluetooth Nodes get connections to neighbor nodes from the Simulation Setup as well as inquiry results and distance estimations.

There are two interfaces to the Simulation Setup. The Controller Interface and the Node Interface.

The Controller Interface is the link from the Controller to the simulation. The controller can just use this interface to communicate with the simulation.

The Node Interface is the link from the Nodes to the simulation. As for the Controller the Nodes are communicating through this interface to the simulation.

This makes it possible to have several different Simulation Setups. For instance, one using random generated data and one using real data. As long as both implement the interfaces nothing changes for the rest of the simulation.

3.3 Positioning Process

The simulation works with 2D coordinates. Each node needs at least three neighbors to calculate its position.

The Controller issues a node to estimate its position. Therefore it gets a connection to the Bluetooth Node from the Simulation Setup. Using this connection, the command to estimate the position is sent to the Bluetooth Node.

The Bluetooth Node will make an inquiry to search in its environment for neighboring nodes. The inquiry is actually directed to the Simulation Setup. The Simulation Setup calculates the inquiry result using the stored information about the simulation. After the Bluetooth Node has received the inquiry result, it opens a communication channel to all nodes in the inquiry result. Using this channel the Bluetooth Node asks the position of its communication partner. Then it gets a distance estimation on the connection. This distance estimation is again provided by the simulation setup. If the Bluetooth Node was able to collect position and distance data of at least 3 neighboring nodes it can calculate its position.

The position data is stored on the local Bluetooth Node. The controller has to collect this data from the node.

4

Implementation

This chapter explains the implementation in more detail.

4.1 Compilation

The file structure of the simulator project is divided in different directories. Netsim is the main directory. It contains the project data required for Visual C++ and the subdirectories bluetooth_sim, gui and res.

The subdirectory bluetooth_sim contains the core of the simulation including the subdirectory libnewmat. libnewmat contains the code for the newmat09 math library.

The subdirectory gui contains the files needed for the GUI. The GUI is just available for the Microsoft Visual C++ project.

The subdirectory res contains some resource files also needed for the GUI.

4.1.1 Visual C++

To compile the Microsoft Visual C++ project open the project file Netsim.dsw with Visual C++. Once the project is opened look for the build icon or choose the menu item from the drop down menu. After the compilation has completed the program can be executed directly from Visual C++. To get further help on simulation commands type 'help' on the command line in the main window.

4.1.2 Linux

The subdirectory bluetooth_sim contains a makefile for the project. To execute the directives in this makefile change to the bluetooth_sim directory and execute the command 'make all'. The simulation will be compiled after a few seconds. To start the simulation execute the command './simulator'. To get further help on simulation commands type 'help' at the simulation prompt.

4.2 Class Implementations

The class hierarchy shown in figure 4-1 corresponds with the simulation concept shown in figure 3-1. The Controller, the Simulation Setup and the Bluetooth nodes are represented as class trees. The oval shapes denote the interfaces.

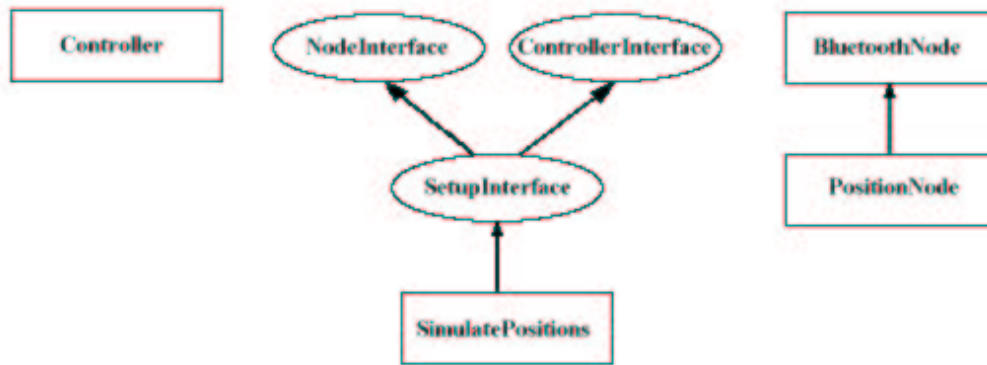


Figure 4-1

The class hierarchy shows the Controller, the Simulation Setup and the Bluetooth Nodes as class trees. The oval shape marks interfaces while the rectangular shape denotes normal classes.

4.2.1 Controller Class

The Controller class is the human interface to the simulation. The controller executes command input and has several methods to return simulation results.

Public methods are:

execute is executing the command passed as a list of strings.

getResultString is returning the result string. The result string describes the result of the last command issued.

getRealPositions is returning a list with the actual positions of the Bluetooth Nodes.

getEstimatedPositions is returning a list with the estimated positions of the Bluetooth Nodes.

getNodeNames returns a list with the names of all Bluetooth Nodes.

4.2.2 SimulatePositions Class

The SimulatePositions class is the core of the simulation. It holds the data to run the simulation. The real positions of the Bluetooth Nodes is generated with random data. This data is used to calculate inquiries and the distance estimations.

The Constructor takes 4 parameters. First parameter is the number of Bluetooth Nodes to be generated. Followed by the field size with the length and width of the simulation area. Finally the error rate for the distance estimation.

The `SimulatePositions` class is implementing the interfaces `NodeInterface` and `ControllerInterface`. This two interfaces are combined in the `SetupInterface`.

Public methods implementing the `NodeInterface`:

estimateDistance takes the names of two Bluetooth Nodes as parameters. The function returns the distance between the two nodes with an error added.

inquiry takes the name of a Bluetooth Node as parameter. The function returns a list with the reachable nodes.

getConnection takes the names of two Bluetooth Nodes as parameter. The function returns a reference of the node to connect.

Public methods implementing the `ControllerInterface`:

getNode takes the name of a Bluetooth Node as parameter. The function returns a reference to the Bluetooth Node.

getNodePosition takes the name of a Bluetooth Node as parameter. The function returns the real position of a Bluetooth Node.

getNodeNames returns a list with the names of all Bluetooth Nodes.

Additional public methods:

setError_Rate takes a floating point variable as parameter. Sets the error rate for the distance estimation.

getError_Rate returns the current error rate for the distance estimation.

setFieldSize takes two floating point variables as parameter. Sets the simulation area to the specified dimensions.

getFieldSize returns the simulation area size.

4.3 *BluetoothNode Class*

The `BluetoothNode` class models the behavior of Bluetooth devices. Basic functions like the `inquiry` is implemented.

Public methods:

inquiry returns a list of reachable Bluetooth Nodes.

getName returns the Bluetooth device name.

getDistanceEstimation takes the name of a Bluetooth Node as parameter. The function returns a distance estimation of the range to the node specified.

connectToNode takes the name of a Bluetooth Node as parameter. The function returns a reference of the specified node.

4.4 *PositionNode Class*

The `PositionNode` class is derived from the `BluetoothNode` class. The class handles the positioning data and the positioning algorithm.

Public methods:

getPosition returns the position of the node.

setExternalPosition takes a position value as parameter. Sets the position of the node to the specified value. The position of the device is not changing until the external position is removed again.

removeExternalPosition the external position is removed from the node. The node will estimate his position next time the command to estimate the position is issued.

calculatePosition the node tries to calculate his position if the position is not set to external.

hasExternalPosition returns true if the position is set to external.

reset is setting the position data to unknown.

5

Summary & Outlook

The basic structure of the simulation is finished. The simulation is working as far as it could be tested.

The first tests show already that the algorithm to calculate the positions is not satisfactory. A new and more complicated algorithm needs to be implemented.

There are some things that still need to be done. A new Simulation Setup for real data needs to be developed. Just experiments with this data will show if the chosen approach in this term work is suitable.

If the approach is suitable it will be interesting to implement a Bluetooth ad-hoc network which is calculating its topology.

Bibliography

- [1] Jan Beutel. Geolocation in a PicoRadio Environment. Departement of Electrical Engineering, ETH Zurich, <http://www.tik.ee.ethz.ch>
- [2] BlueTooth Special Interest Group. <http://www.bluetooth.com>.
- [3] Chris Savarese, Jan M. Rabaey and Jan Beutel. Locationing in Distributed Ad-hoc Wireless Sensor Networks. May 2001.
- [4] Axis OpenBT Stack. <http://developer.axis.com>.
- [5] Newmat09 Library <http://webnz.com/robert>.
- [6] Linux Operating System. <http://www.linux.org>.