

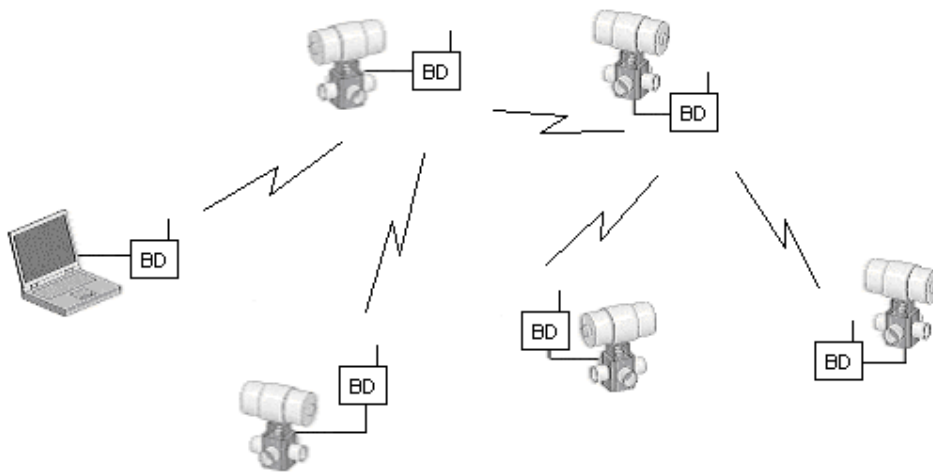


connectBlue

*Lund Institute of Technology*

# Self-configuring Bluetooth<sup>TM</sup> Networks

Igor Gurovski & Velimir Karadzic



Bluetooth is a trademark of Bluetooth SIG Inc.

## **Abstract**

According to the Bluetooth specification v1.1, several smaller Bluetooth networks can be linked together in a network topology called a *scatternet*. The Bluetooth devices available on the market today do not support this feature but hopefully very soon there will be Bluetooth chips that make it possible. Then large networks involving theoretically an infinite number of devices could be created. However, each of these devices has to be configured in order to establish the connections.

The Bluetooth specification does not specify how these connections could be established automatically. It therefore would be interesting to develop a method where the units investigate their surrounding and form a scatternet in an optimal way. The aim of this master's thesis is to find an algorithm for this purpose.

Radio link's vulnerability is the reason why only the best link should be prioritised. A scatternet-forming algorithm that builds on Prim's algorithm was developed and implemented in order to fulfil the requirements. Special hardware was designed for emulating of true scatternet. The algorithm was successfully tested in a small demonstration.



## ***Preface***

This report is the result of the master's thesis carried out at connectBlue AB in Malmö, Sweden. It is also a part of our Master of Science degree at Lund Institute of Technology (LTH).

We would like to thank Per Nilsson and Mats Andersson at connectBlue AB for being our supervisors, and Associate Professor Mats Cedervall at LTH for being our examiner.

Malmö, December 2002

## Table of Contents

<b>1. Introduction</b>	<b>1</b>
<b>1.1 Problem Description</b>	<b>1</b>
<b>1.2 Thesis Objective</b>	<b>1</b>
<b>1.3 Report Structure</b>	<b>2</b>
<b>2. Bluetooth Wireless Technology</b>	<b>3</b>
<b>2.1 Piconet</b>	<b>4</b>
<b>2.2 Scatternet</b>	<b>4</b>
<b>3. The Serial Port Adapter (SPA)</b>	<b>8</b>
<b>3.1 The Serial Port</b>	<b>8</b>
<b>3.2 Description of the Serial Port Adapter</b>	<b>8</b>
<b>3.3 Configuring a Serial Port Adapter</b>	<b>10</b>
<b>3.4 Limitations of the Serial Port Adapter</b>	<b>11</b>
<b>4. Larger Bluetooth Networks</b>	<b>13</b>
<b>4.1 Configuration of larger Bluetooth networks</b>	<b>13</b>
<b>4.2 Self-configuring Bluetooth networks</b>	<b>14</b>
<b>5. Scatternet Forming Algorithms</b>	<b>15</b>
<b>5.1 Requirements on the Scatternet Forming Algorithms</b>	<b>15</b>
<b>5.2 The Dijkstra's Algorithm</b>	<b>15</b>
<b>5.3 A Bluetooth Scatternet Formation Algorithm</b>	<b>16</b>
<b>5.4 Tree Scatternet Formation (TSF)</b>	<b>16</b>
<b>5.5 Prim's Algorithm</b>	<b>17</b>
<b>5.6 Conclusions</b>	<b>18</b>
<b>6. AC unit</b>	<b>19</b>
<b>6.1 The Hardware of the AC unit</b>	<b>19</b>
<b>6.2 The Software of the AC unit</b>	<b>20</b>
<b>7. Demonstration</b>	<b>25</b>
<b>8. Result</b>	<b>27</b>
<b>9. Future Work</b>	<b>28</b>
<b>A. References</b>	<b>29</b>
<b>B. Acronyms</b>	<b>30</b>
<b>C. The Bluetooth Stack</b>	<b>31</b>
<b>C.1 Radio</b>	<b>32</b>
<b>C.2 Baseband</b>	<b>33</b>

<b>C.3 Link Manager</b>	<b>34</b>
<b>C.4 Host Controller Interface</b>	<b>34</b>
<b>C.5 Logical Link Control and Adaptation Protocol</b>	<b>35</b>
<b>C.6 RFCOMM Protocol</b>	<b>36</b>
<b>C.7 Service Discovery Protocol</b>	<b>36</b>
<b><i>D. Bluetooth profiles</i></b>	<b>38</b>
<b>D.1 Generic Access Profile</b>	<b>39</b>
<b>D.2 Serial Port Profile</b>	<b>40</b>
<b>D.3 Dial-up Networking Profile</b>	<b>40</b>
<b>D.4 LAN Access Profile</b>	<b>41</b>
<b>D.5 Personal Area Networking Profile</b>	<b>41</b>



The self-configuration also means that no extra job with configuring is needed. A simple command, e.g., a button starts the configuration and after some time there is a unique way between the devices. The user does not have to know anything about how to set up a network. The result of this master's thesis makes it happen automatically.

### 1.3 Report Structure

- Chapter 1** gives a concise description of the main issues in the master's thesis as well as the thesis objectives.
- Chapter 2** presents an overview of the Bluetooth technology according to the Bluetooth specification v1.1. A detailed description of scatternet is given.
- Chapter 3** describes the Serial Port Adapter qualities, how it works, how it can be used and which limitations are present.
- Chapter 4** explains the background of the master's thesis and the final goal is described.
- Chapter 5** gives some alternatives for scatternet forming algorithms and arguments for the chosen algorithm.
- Chapter 6** describes the AC-unit hardware and the software qualities.
- Chapter 7** gives a description of the demo developed in the purpose to show the result of the master's thesis.
- Chapter 8** contains the result developed in the master's thesis and potential future work.
- Appendix A** is references.
- Appendix B** is acronyms.
- Appendix C** gives a description of the Bluetooth stack and the protocols it involves.
- Appendix D** summarises the most important Bluetooth profiles.

## **2. Bluetooth Wireless Technology**

Bluetooth, named after Denmark's first Christian king Harald Blåtand, is the name of a technology specification for low-cost, short-range radio links between PCs, mobile phones and other computing and electronic devices [4]. The technology was originally developed by Ericsson. The Ericsson inventors understood that the technology was more likely to be widely accepted, and thus could be more powerful, if it was adopted by an industry group that produce an open, common specification. The Bluetooth SIG (Special Interest Group) is an industry group consisting of leaders in the telecommunications and computing industries that are driving development of the Bluetooth technology and bringing it to market. The founding companies are Ericsson, Intel, IBM, Nokia and Toshiba. Today over 2000 companies have signed the Bluetooth adopter's agreement and are members of the Bluetooth SIG [6].

There are already a couple of ways to get around using wires. One is to carry information between components via beams of light in the infrared spectrum. Infrared refers to light waves of a lower frequency than human eyes can receive and interpret. Infrared is used in most television remote control systems, and with a standard called IrDA (Infrared Data Association) it is used to connect some computers with peripheral devices.

Bluetooth is intended to get around the problems that come with both infrared and cable synchronizing systems. From the user's point of view, there are three important features of Bluetooth:

- It is wireless. When you travel, you do not have to worry about keeping track of a briefcase full of cables to attach all of your components. Bluetooth devices do not need line of sight as IrDA devices for proper communication.
- You do not have to buy Bluetooth enabled devices from the same manufacturer. Bluetooth guarantee interoperability between different manufacturers.
- It is inexpensive.

Bluetooth communicates on a frequency of 2.45 gigahertz, which has been set aside by international agreement for the use of industrial, scientific and medical devices. A number of devices that you may already use take advantage of this same radio-frequency band. Baby monitors, garage-door openers and the newest generation of cordless phones all make use of frequencies in the ISM band. Making sure that Bluetooth and these other devices do not interfere with one another has been a crucial part of the design process.

One of the ways Bluetooth devices avoid interfering with other systems is by sending out very weak signals of 1 milliwatt. By comparison, the most cell phones can transmit a signal of 1-2 watts. The low power limits the range of a Bluetooth device to about 10 meters, cutting the chances of interference between your computer system and your portable telephone. Because of the signals high frequency, the walls in your house will not stop the Bluetooth signal, despite the low power consumption. This makes the Bluetooth standard very useful for controlling several devices in different rooms.

## 2.1 Piconet

Devices connected via Bluetooth technology connect in an ad hoc fashion. When two Bluetooth devices connect they form a network, which in the Bluetooth standard is called a piconet [8]. A piconet starts with two connected devices, such as a portable PC and cellular phone, and may grow up to eight connected devices. All Bluetooth devices are peer units and have identical implementations. However, when establishing a piconet, one unit will act as a master and the other(s) as slave(s) for the duration of the piconet connection. A slave can only communicate with its master in the piconet. All devices in a piconet share the same frequency hop sequence, which is calculated using the master's clock and Bluetooth address [1]. The master controls how the total available bandwidth is divided among the slaves, by deciding how often and when to communicate with each slave, using a Time Division Multiplexing (TDM) scheme.

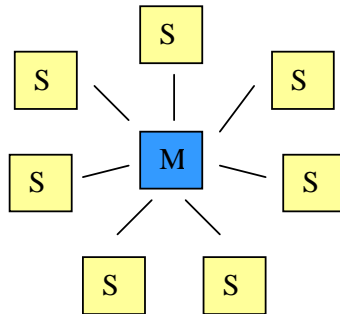


Figure 2.1: A piconet with a master and the maximum of seven active slaves.

## 2.2 Scatternet

Multiple piconets may cover the same area. Since each piconet has a different master, the piconets hop independently, each with their own channel hopping sequence as determined by the respective master. If multiple piconets cover the same area, a unit can participate in two or more overlaying piconets. When this happens a scatternet is created [8][5]. Though the specification limits the number of slaves in a piconet to seven, the use of scatternet can increase the number of network members and make the coverage area larger. When a device is present in more than one piconet, it must time-share. This means that the device must spend a few slots on one piconet and a few slots on the other. A Bluetooth unit can act as a slave in two piconets, but only as a master in a single piconet. See Figure 2.2.

In order to make communication between piconets possible, the devices that are acting as intermediate devices between piconets, need to be able to forward packets between the nets.

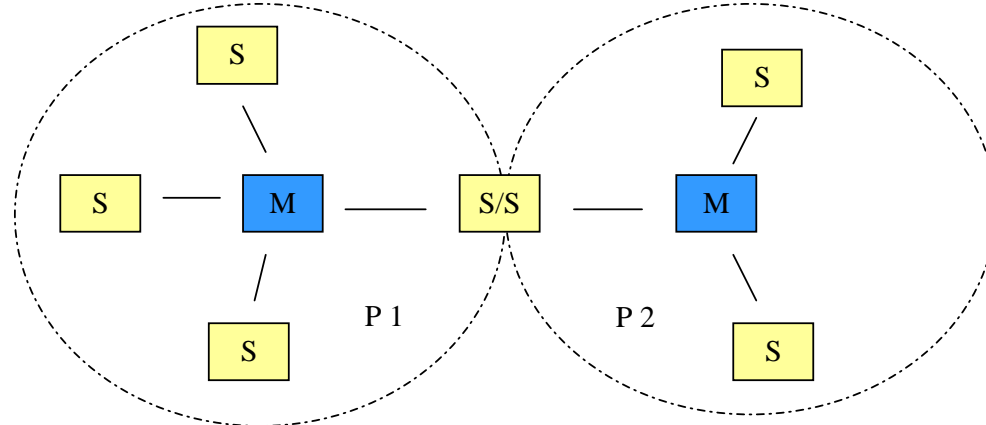


Figure 2.2: A scatternet consisting of two piconets.

According to the Bluetooth specification v1.1 [6], a Bluetooth unit can act as a slave in two piconets, but only as a master in one piconet. If a master would be a master in two piconets, these piconets would be synchronized and use the same hopping sequence. In other words these two piconets would be one and the same piconet. In Figure 2.3  $M_2$  is paging, i.e., connecting to  $S_1$  and a scatternet is formed.

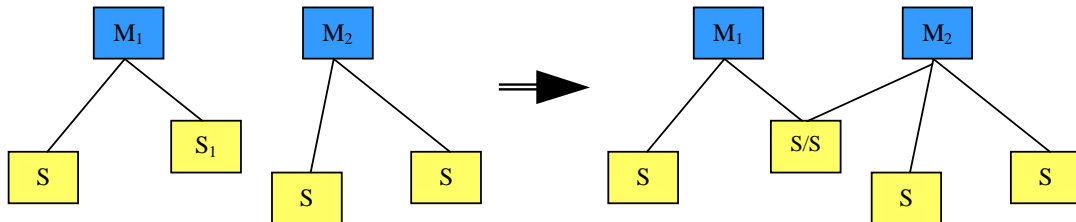


Figure 2.3: A Bluetooth device acts as a slave in two piconets.

A master or a slave can become a slave in another piconet by being paged by the master of this other piconet. See Figure 2.4. For example, master  $M_1$  in Figure 2.3 could page master  $M_2$ . Then  $M_2$  would have a double role: master in one piconet and a slave in the other piconet.

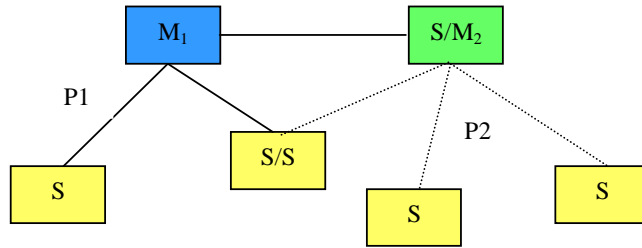


Figure 2.4: A Bluetooth master device becomes a slave in another piconet.

According to the specification, a unit participating in one piconet can page the master or slave of another piconet. See Figure 2.5. For example, one of the units in M<sub>2</sub>'s piconet pages and becomes a master, here M<sub>3</sub>, over a unit in M<sub>1</sub>'s piconet.

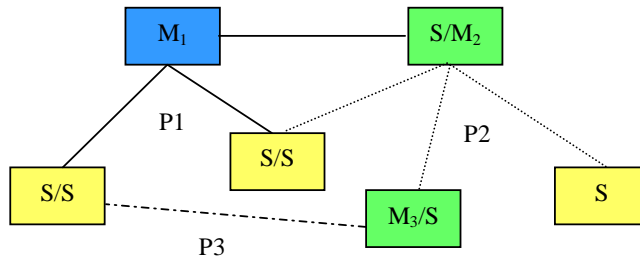


Figure 2.5: A Bluetooth unit pages another unit, outside its own piconet.

Sometimes it is desirable for a Bluetooth unit to change its master/slave role. The most common cases are following:

1. A unit pages the master of an existing piconet and wants to join the piconet. Since, by definition, the paging unit initially is the master, a master/slave switch is necessary. See Figure 2.6.

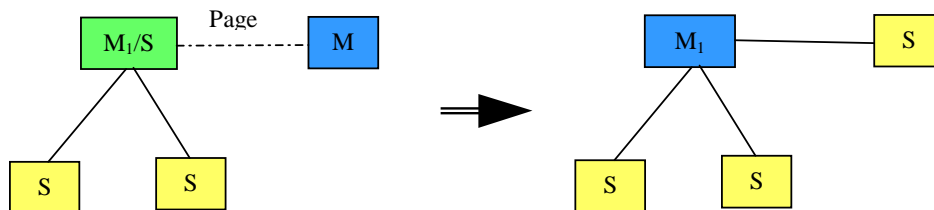


Figure 2.6: A unit pages a piconet's master and then makes a master/slave switch.

2. The slave  $S_2$  in the existing piconet wants to set up a new piconet. The  $S_2$  slave first makes a master/slave switch with master  $M_1$ .  $S_2$  have now become  $M_2$  and can establish its own piconet. See Figure 2.7.

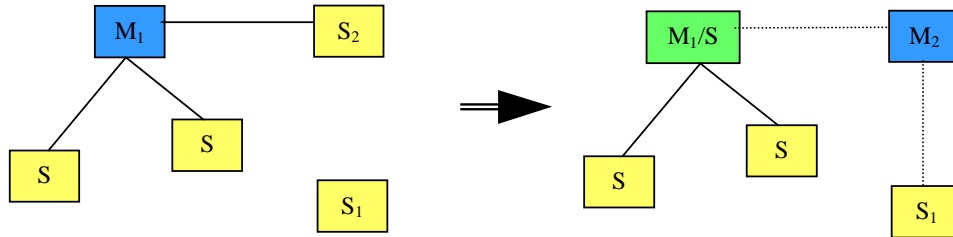


Figure 2.7: A unit wants to set up a new piconet.

3. A slave wants to fully take over an existing piconet, i.e., the switch also involves transfer of other slaves of the existing piconet to the new piconet. Clearly, this can be achieved by letting the new master, setup a completely new piconet through the conventional paging scheme. However that would require individual paging of the old slaves and thus take unnecessarily long time. Instead, letting the new master utilize timing knowledge of the old master is more efficient. As a consequence of the master/slave switch, the slaves in the piconet have to be transferred to the new piconet, changing their timing and their hopping scheme. See Figure 2.8.

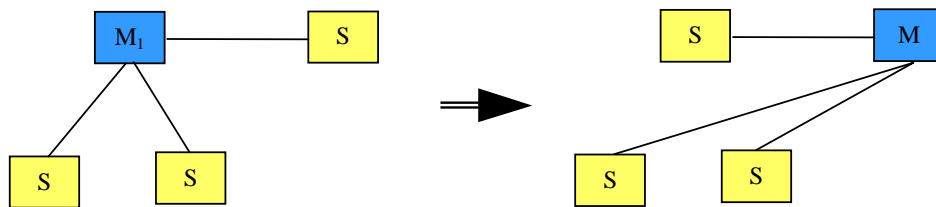


Figure 2.8: Transferring of slaves to the new piconet, after the master/slave-switch.

### 3. The Serial Port Adapter (SPA)

The Serial Port Adapter is the product used in the master's thesis *Self-configuring Bluetooth networks* [19]. Before a description of its original qualities is given, several concepts should be clarified.

#### 3.1 The Serial Port

The *serial port*, also called communication (COM) port, has been an integral part of various systems for more than 30 years. For example, most PC's today have one or two of these ports. As the word serial indicates, the sent data is serialized, which means that data is transmitted one bit at a time. The most common serial port has a 9-pin connector. See figure below. There are also some older 25-pin connectors.

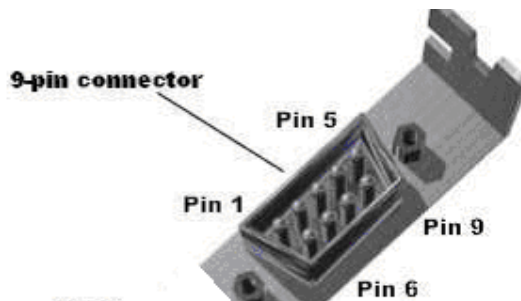


Figure 3.1: 9-pin connector

The 9-pin connector pins are:

1.Carrier Detect	4.Data Terminal Ready	7.Request To Send
2.Receive Data	5.Signal Ground	8.Clear To Send
3.Transmit Data	6.Data Set Ready	9.Ring Indicator

A program running on a certain system can send digital data to the transmit pin (output) and receive bytes from the receive pin (input). The other pins are for control purposes and ground.

To function properly, serial ports use a special controller chip called *Universal Asynchronous Receiver/Transmitter (UART)*. The UART chip takes the parallel output of a device's system bus and transforms it into serial form for transmission through the serial port. The electrical specification of the serial port is contained in the Electronics Industry Association (EIA) *RS232C* standard.

#### 3.2 Description of the Serial Port Adapter

The Serial Port Adapter developed by connectBlue AB is shown in Figure 3.2. Figure 3.3 shows the same product without protective cover.



Figure 3.2: Serial Port Adapter used in Self-configuring Bluetooth networks project.



Figure 3.3: Serial Port Adapter without protective cover.

The SPA allows any device, with an RS232 port, to communicate wirelessly with other devices containing an RS232 port. It is commonly used in scenarios that involve fixed units.

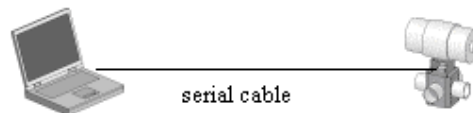


Figure 3.4: A PC connected to a loop controller by a serial cable.



Figure 3.5: A Bluetooth enabled PC connected to a loop controller.

Both the PC and the loop controller in Figure 3.4 are equipped with an RS232 port. Communication between these two devices can be enabled using a serial cable.

In Figure 3.5, the same goal is achieved using two Serial Port Adapters. Each device is connected to an SPA through its RS232 port and Serial Port Adapters communicate wirelessly. An SPA is set either in packet or data mode. An SPA in packet mode handles commands sent to it. When in data mode, an SPA is transparent for all incoming data, i.e., the data coming via Bluetooth is forwarded to the application UART and vice versa.

Perhaps, the first reaction of the reader is: *If wanted communication is enabled by a serial cable, is not it a waste of money to buy two SPAs and then need to connect them to the devices using serial cables, to achieve the same goal?*

The answer is yes, if a cable can connect the two devices without causing any trouble.

Unfortunately, there are various scenarios where it would be much more expensive and sometimes even impossible to connect two devices by a serial cable. Imagine that there is a wall between the PC and the loop controller in Figure 3.4. Radio waves attenuate but hopefully reach the device on the other side of the wall. Thus, a device does not have to be physically reachable to be controlled. Another possible industrial application scenario that points out the advantage of using SPA is the following:

Several Bluetooth enabled loop controllers, spread in a factory, should be controlled. A person with a Bluetooth enabled PC or PDA can move around and connect wirelessly up to seven of them simultaneously, from a distance of about 10 meters. A multi-point connection eases the work.

An SPA quality that deserves to be emphasised is that no software installation and no changes in the existing system (PC and loop controller) is needed.

### **3.3 Configuring a Serial Port Adapter**

The PC in Figure 3.4 is physically connected to the loop controller by a serial cable. However, it is not the only condition that has to be fulfilled for correct transmission of the data. The serial settings for the selected com-port have to be specified. Of course, these settings should be the same on both devices. The parameters that specify an RS232 port are: bits per second, data bits, parity, stop bits and flow control. Once, a cable connects the devices and the serial port settings are specified, bi-directional communication can start.

*But, how does it work if SPAs are used, as in Figure 3.5?*

The configuration of the serial ports settings on the PC, the loop controller and the SPAs is still inevitable. Besides, the SPAs connected to the PC and the loop controller have to know which device to send incoming data to. This configuration is done using a tool called *Windows SPA wizard*. A customer gets this software application when purchasing an SPA. A PC with the wizard running connects to the SPA via the RS232 port and configuration can be performed. The wizard sets the SPA in packet mode and various parameters can be set. One of them is called *default remote peer*. It is the Bluetooth address of the device we want to connect to. In Figure 3.5, the remote peer of the SPA connected to the PC is the SPA connected to the loop controller.

Thus both configuration of serial ports parameters and configuration of SPAs are necessary before bi-directional communication can start.

Using Windows wizard is not a complicated task for a person who knows the procedure. Each unit can be configured in a couple of minutes. The time it takes to perform configuration of Serial Port Adapters is usually acceptable.

### **3.4 Limitations of the Serial Port Adapter**

A Bluetooth device's (SPA) ability to form a scatternet is one of the most important preconditions in this master's thesis. Therefore, an investigation that compares the Serial Port Adapter's real scatternet characteristics and the Bluetooth specification characteristics is done. If the SPAs could be connected according to the scenarios described in the Section 2.2, then we can say that an SPA supports scatternet. The results of connecting attempts are given below.

Figure 2.2 in Section 2.2 shows a case in which a Bluetooth device acts a slave in two piconets. Testing to connect the SPAs in this manner fails for the following reason:

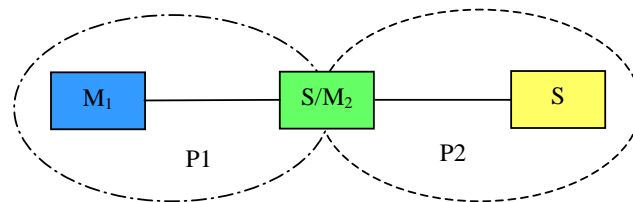
Once, an SPA becomes a slave in a piconet, it is both non-discoverable and non-connectable. It means, when the master device  $M_2$  does device discovery, i.e., it searches for all devices within its range, it cannot find the device (SPA) that already is a slave in  $M_1$ 's piconet.

Another important restriction of the available Serial Port Adapters is that an SPA, which is a master in one and a slave in another piconet, always behaves as a slave. It means, no device can find or connect to an SPA that is both master and slave. See Figure 2.4 in Section 2.2.

A test shows that the scenario, described in Figure 2.5, is also impossible to perform. The reason is that the SPAs do not allow a slave to make an inquiry or connect to other Bluetooth devices.

Except of the restrictions concerning finding and connecting of devices, a test revealed an additional limitation:

A node (SPA), which is master in one and slave in another piconet, crashes when data comes from the master in the other piconets. See the figure below:



*Figure 3.6: When  $M_1$  sends data to its slave, the slave crashes.*

Summarized, these are the following limitations concerning an SPA's ability to support scatternet:

1. A slave is always non-discoverable and non-connectable
2. A master/slave cannot receive data from another master
3. A master/slave always behaves as a slave
4. A slave can neither search for nor connect to other Bluetooth devices

Unfortunately, *the Serial Port Adapter or more exactly its radio chip, which is about to be used in this master's thesis, does not support scatternet*. This fact complicates the further work and has to be solved. How it is done, is described in Chapter 5 (AC-unit).

## 4. Larger Bluetooth Networks

As mentioned before, linking two or more piconets forms a scatternet. Section 2.2 Scatternet explains main principles of this context. In theory, a scatternet could involve an infinite number of SPAs.

For the reasons given in Section 3.4 *Limitations of the Serial Port Adapter*, SPAs still do not support forming of a scatternet. Hopefully soon the Serial Port Adapter will contain a Bluetooth radio chip that fully supports this feature. Then, with some modifications made in the firmware, a lot of application scenarios involving many SPAs will be possible and hence wider surfaces could be covered. The figure below shows a possible scenario.

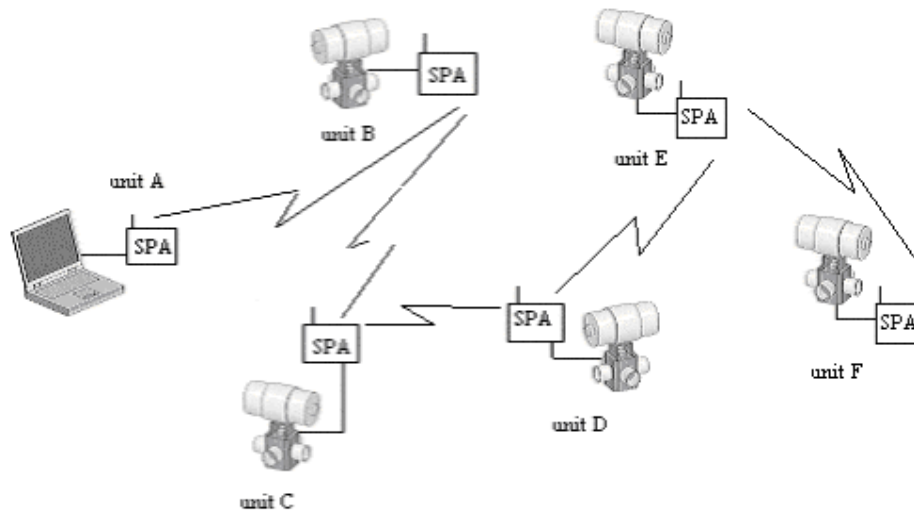


Figure 4.1: An application scenario including SPAs that support scatternet

The PC in Figure 4.1 is the central unit of the network. The data from units B to F ought to be gathered and handled in the unit A. The PC cannot reach all of the units directly, but a scatternet can be set up. A unique way between all units is established and used for transmission of data.

### 4.1 Configuration of larger Bluetooth networks

Manual configuration of the networks, as the one shown in Figure 4.1, can be cumbersome for several reasons. First of all, there is still a need for a technician that can handle the wizard. Configuration of many SPA units can take unacceptable long time and sometimes it can be difficult to physically approach the SPAs. Besides a technician chooses *default remote peer* for an SPA. Usually he does not know much about the quality of the link between two chosen SPA units. This can cause bad link quality between the units. Perhaps there is some better alternative way for the SPAs to reach each other.

For example, in Figure 4.1 the technician chooses to configure SPA unit A so, that it connects to the unit B directly. This occurs to be a very slow link. It would be much more effective if the unit A should know that there is an alternative, better way to connect to the unit B. In Figure 4.1, it would be possible by connecting to the SPA unit C, which is already connected to the B.

Every time a new SPA unit is about to be added to the already configured network, a technician is needed and the problem to choose which SPA to connect to persists. When all Serial Port Adapters, participating in a network, have been configured, there is a unique way between each device participating in the network. Figure 4.1 shows a possible network topology after configuration. After the network has been configured, communication between involved devices can start.

Summarized, manual configuration of larger networks has following disadvantages:

- Manual configuration takes time and thus raises the costs.
- Manual configuration can be hard to perform depending on SPAs physical location.
- Manual configuration does not guarantee that the best unique way between involved SPA units is chosen.

## 4.2 Self-configuring Bluetooth networks

The goal of this master's thesis was to enable packet forwarding in larger networks and to eliminate the disadvantages caused by manual configuration.

The first step was to find out if the Bluetooth specification v 1.1 says anything about how to solve these problems. The PAN profile, described in appendix D.5, sounded as the closest and only interesting solution. That is the reason why it has been nearly examined. The PAN profile enables the forwarding of packets between two PAN users. However, even with Bluetooth devices supporting the PAN profile, there still persists a need for a scatternet-forming algorithm and the PAN profile does not say anything about that.

The following scenario becomes possible by using the algorithm developed in this thesis:

Serial Port Adapters, which support scatternet, are connected to hosts, e.g., PC or loop controllers, through their com-port. The com-port parameters are still configured manually. Pressing a button on one of the SPAs starts the self-configuration. Each SPA measures the link quality to its neighbours and connects to the best one. The one that became connected last continues the procedure until all SPAs within radio range are connected. Finally, there is a unique way between the units and self-configuration is finished. The connection tree can look like the one in Figure 4.1. There is no need for a wizard, all approachable nodes are connected and only the best links are used.

*Any Bluetooth devices, i.e., not only SPAs, can use the algorithm in order to form the optimal connection tree.*

## 5. Scatternet Forming Algorithms

The AC-unit described in the next chapter gives an opportunity to create a scatternet, but there is still a question about how the network topology should look like. It means that an optimal way for creating the connection tree between the involved Bluetooth devices has to be found. Several scatternet forming algorithms, including the one implemented in the thesis, are described in this chapter.

Before a description of each algorithm is given, some requirements should be presented.

### 5.1 Requirements on the Scatternet Forming Algorithms

The algorithm has to involve all Bluetooth devices that want to participate in the network and that are within the radio range. It has to investigate which devices that are capable and allowed to participate in the wanted scatternet and which are not.

It also has to be executed independently on each one of the involved Bluetooth devices. Any kind of a central Bluetooth device should be avoided because it makes the network vulnerable. If the central unit disappears, the entire network would disappear. The fact that all Bluetooth devices, we are about to connect, are not always within radio range of the central unit ratifies this requirement too.

The algorithm has to be able to handle that devices can be added or removed from the network.

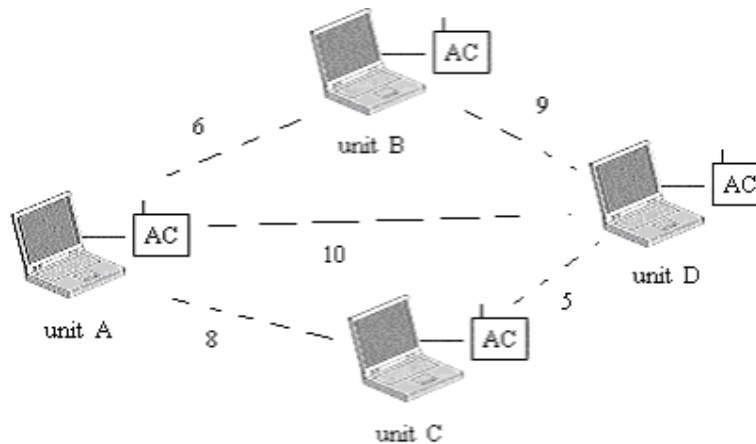


Figure 5.1: A network scenario with known link costs between the devices

### 5.2 The Dijkstra's Algorithm

A scatternet-forming algorithm that builds on Dijkstra's algorithm could be developed [2]. Dijkstra's algorithm finds the shortest path from one to all other nodes in the network. Applied on the devices in Figure 5.1, the following scenario becomes possible: Unit A starts the configuration and it calculates the shortest path to all reachable nodes. The connection tree gets the shape shown in Figure 5.2.

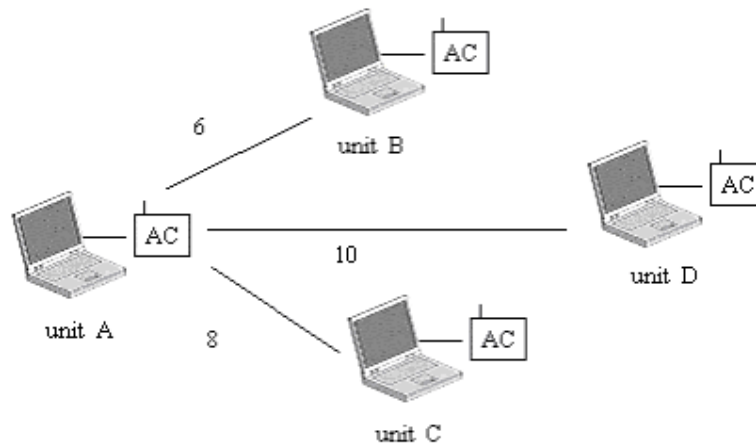


Figure 5.2: The network topology tree established using Dijkstra's algorithm.

A unique way between the involved devices is established, but unfortunately the bad quality link between the units A and D is chosen.

Bluetooth uses the radio medium, which makes links vulnerable. It is the reason why high quality links are preferred. It is so important that even some extra hops can be made in order to avoid the bad links. An algorithm that by all means excludes the unreliable links had to be found and that is why Dijkstra's algorithm is not chosen.

### 5.3 A Bluetooth Scatternet Formation Algorithm

This solution is developed by Ching Law and Kai-Yeung Siu and is explained in details in the work *A new Bluetooth Scatternet Formation Protocol* [10].

According to the developers it is a very fast and effective algorithm. It supports cases when devices arbitrary arrive to an already formed network.

The reason why it has not been an option for this master's thesis is that all involved Bluetooth devices have to be within radio range to each other.

### 5.4 Tree Scatternet Formation (TSF)

Tan, Miu, Gutttag and Balakrishnan propose a scatternet formation algorithm in *Forming Scatternet from Bluetooth PAN* [11]. It is called the Tree Scatternet Formation and it works in the following way:

When the scatternet formation starts, a Bluetooth device can be in one of two different states: INQUIRY or INQUIRY\_SCAN. It switches between these two states in a randomness fashion. When a device receives a successful Inquiry response both devices perform the baseband processes: Page and Page Scan respectively. After a successful Page the two devices have established a connection. The device acting as a master becomes the root and the one acting as a slave becomes the leaf. If a root joins the tree as a child, it gets the master/slave role.

Unlike the previous described algorithm, TSF can form a scatternet including devices that are out of range to each other. Unfortunately, it does not say anything about link quality. It means that the bad links can be chosen instead of the good ones. This is unacceptable for this thesis goal and it excludes this algorithm.

## 5.5 Prim's Algorithm

To explain how Prim's algorithm works, we use the scenario shown in Figure 5.1 [7][9].

The user chooses unit A to start the self-configuration. The A unit does inquiry, i.e., it searches for all discoverable Bluetooth devices within its radio range, and finds the units B, C and D. Then the unit A measures the link quality to these three devices and puts all information (source, destination, cost) in a table (A, B, 6 ; A, C, 8 ; A, D, 10). Now, unit A connects the link with lowest cost, i.e., unit B, and sends the updated table (A, C, 8 ; A, D, 10) to this unit.

The connection between the units A and B is final.

Unit B now continues the procedure by searching for new devices (inquiry), measuring the link quality and then updating the table with the new links. Unit B sees that the new link from B to D is cheaper than the already existing link to D. Therefore the link from A to D is removed from the table (A, C, 8 ; B, D, 9). Unit B returns the new table to unit A since it has the cheapest link. When unit A gets the table it connects to unit C.

The connection between the units A and C is final.

It is now unit C's turn to continue the configuration by first performing an inquiry and then updating the table (B, D, 9). After updating the table (C, D, 5), unit C finds its own link to D cheapest and connects to this unit.

The connection between the units C and D is final.

The connected units cannot be found during self-configuration and that is the reason why unit D, after the performed inquiry, does not find any device. The table, unit D has received, is empty and the D sends a message to all connected devices that configuration is finished. The connections shown in Figure 5.3 are established.

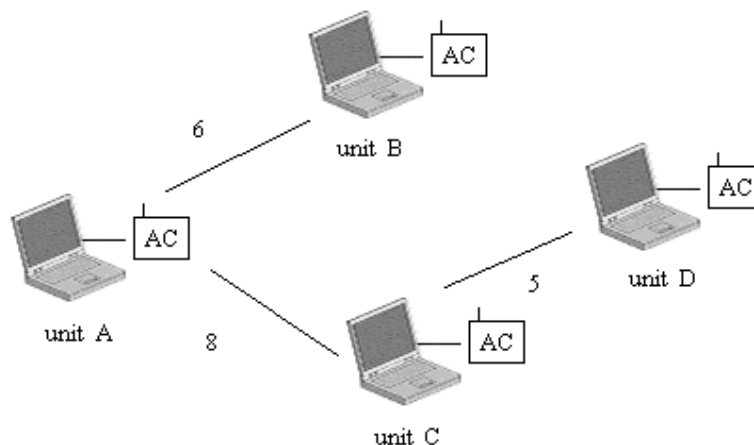


Figure 5.3: The network tree that is established when using Prim's algorithm

After being compared with other algorithms, the Prim's algorithm is found to be the best choice to build the scatternet-forming algorithm on. It defines a unique way between the Bluetooth devices in the optimal manner for the needs of this thesis. It means that the cheapest way to connect *all* nodes in a network is found. The network topology becomes as robust as possible by following this algorithm when connections are about to be established.

## **5.6 Conclusions**

It is not a simple task to find an optimal scatternet-forming algorithm. In some applications the time it takes to set up a network is crucial and has to be low, in others the number of the hops between two devices has to be minimized. The link quality between the devices is often a very important factor. Even the location of the devices that should be connected affects the choice of the algorithm. A good topology choice can only be made after simulations and experiments are performed.

The application that should be covered by this master's thesis requires the qualities offered by Prim's algorithm. That is the reason why the Serial Port Adapters are modified to follow this algorithm during the procedure of automatically setting up a network. The implemented algorithm, build on Prim's idea, is general. It means that it can be used for automatically setting up the networks containing arbitrary Bluetooth devices.

## 6. AC unit

After the investigation described in Section 3.4 *Limitations of the Serial Port Adapter*, it is clear that the radio hardware in the available Serial Port Adapters does not support scatternet. It was the first problem that had to be solved.

A unit that can act master in one and slave in another piconet is needed. As known, an SPA does not manage the simultaneous roles but it works perfectly as only master or slave. To let one SPA act as a master while another one acts as a slave and then somehow make them interact should meet the requirements. For this purpose, a device called *Auto Configuring (AC) unit* is designed. See Figure 6.1.

### 6.1 The Hardware of the AC unit

An AC unit consists of two SPA units. The left one in Figure 6.1 is in data mode and it is unchanged, i.e., it has original SPA firmware. All data coming in via Bluetooth to the left SPA, is forwarded to the modified SPA's internal UART and vice versa. The right SPA is modified and runs software developed in this master's thesis.

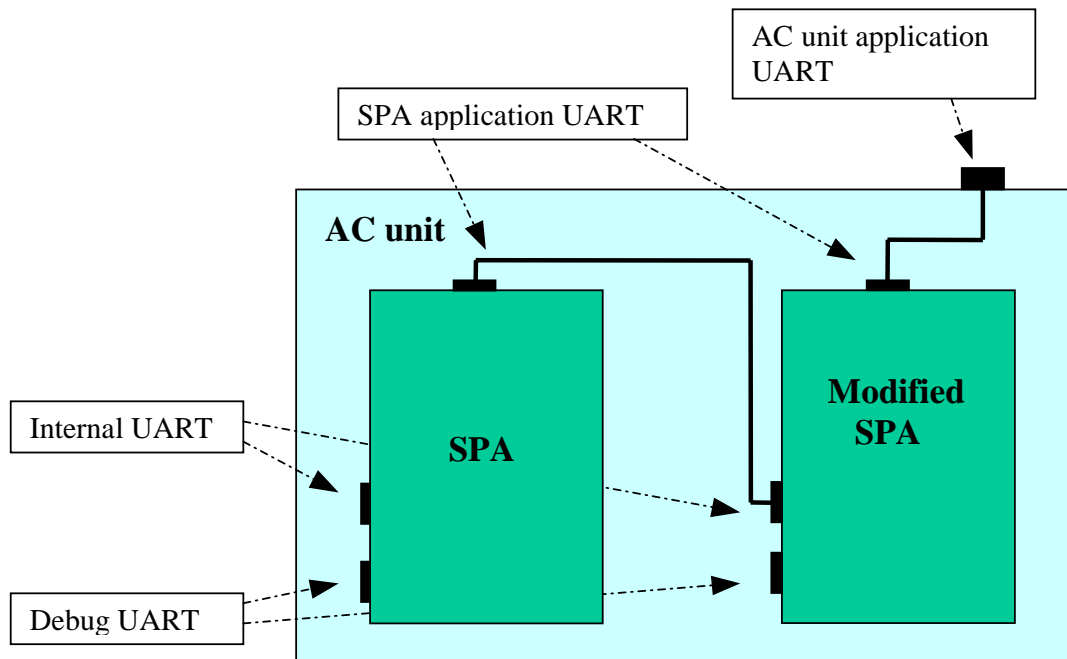


Figure 6.1: AC unit, developed for the scatternet formation purpose.

Every Serial Port Adapter contains three different UARTs: application, internal and debug UART. The application UART is attached through an RS232 chip to a 9-pin connector. See Figure 3.1. The internal UART on the modified SPA is connected to the application UART on the unchanged SPA, which enables communication between the SPAs. The modified SPA's application UART is the AC unit's application UART.

The idea is that the unchanged SPA can act as a slave in one piconet, while the modified SPA can have a master role in another. This is one of the wanted qualities for supporting scatternet. Figure 6.2 shows a scatternet using AC units.

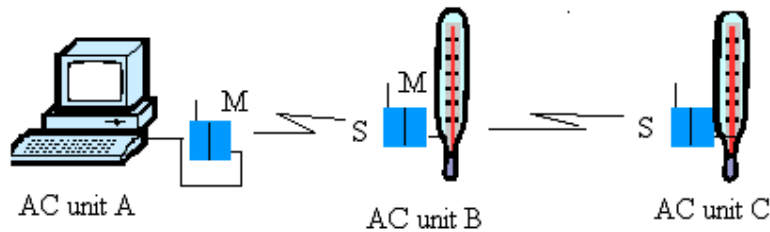


Figure 6.2: AC units enable scatternet.

In the scenario shown in Figure 6.2, unit C is out of range for the A unit. A piconet involving all 3 devices cannot be formed with unit A acting as master. Data communication between the PC and the temperature regulators is desired and could be enabled in the following way:

The PC and the temperature regulators are each connected to the AC-unit's application UARTs. The modified SPA (right unit) of A connects as master to the unchanged SPA of B (left unit), which becomes slave. The modified SPA of B connects as master to the unchanged SPA of C and the configuration is done.

All data sent from the PC is now sent over Bluetooth to the left SPA of unit B. This SPA is in data mode and just forwards incoming data to its application UART, which is connected to the internal UART of the right unit of B. Then, the B unit forwards the data both on its own application UART and over Bluetooth link to unit C. Bi-directional communication is enabled and AC devices support scatternet.

*When the Serial Port Adapter contains radio hardware that supports scatternet, the left unit of an AC device will be unnecessary. Only small changes in the code that runs on the modified SPA will be enough to perform the wanted task.*

## 6.2 The Software of the AC unit

The hardware of the AC unit is described in the previous section. Except for this specially designed solution, it is also required to develop software that will run on these devices. This software has to fulfil the main goal of the thesis, i.e., it has to enable self-configuration and packet forwarding of the AC units. Besides, an AC unit has to offer the same functionality as a Serial Port Adapter does. Of course, only as small as possible modifications of the SPA's original firmware should be made. How these requirements are fulfilled is described in the following:

The left unit in an AC device, as already mentioned, is a common Serial Port Adapter. It means that the software running on it is unchanged. The source code is written in ANSI C, where the term class is not used. Although in order to simplify the description this term is used anyway. The class diagram that shows an SPA's software structure is shown in the Figure 6.3.

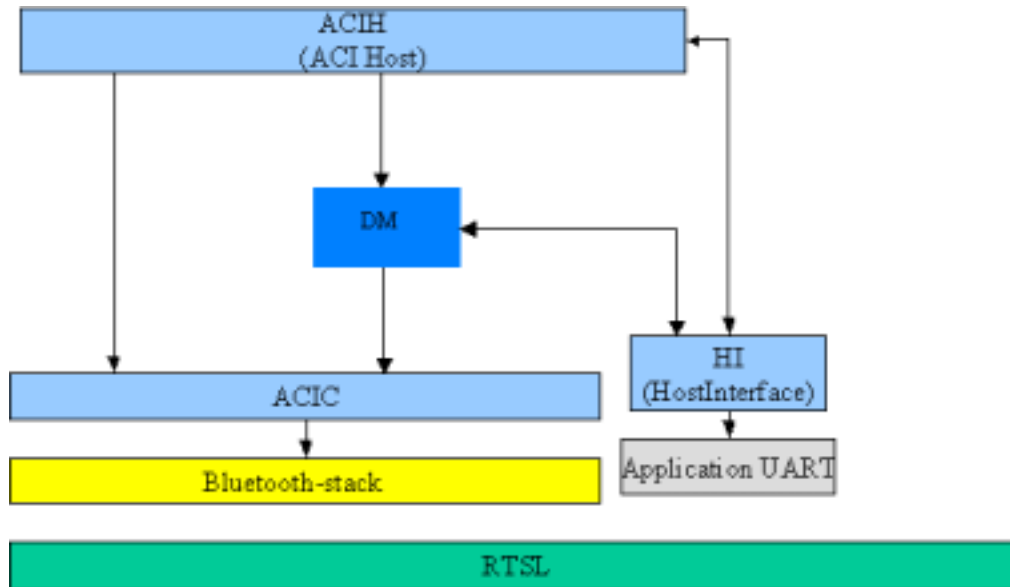


Figure 6.3: The class diagram of the Serial Port Adapter.

The classes shown in Figure 6.3 interact in a way that makes a Serial Port Adapter function properly. A more detailed description of the Serial Port Adapter software is not given here because it is out of scope for this master's thesis.

The right unit in an AC device is a modified SPA. This is the more intelligent unit that enables the goal of this master's thesis. The class diagram for this unit has the structure shown in Figure 6.4.

The class AutoConfig is the software developed for the purpose of this master's thesis. To understand how this class is implemented, some knowledge about underlying classes is needed. As Figure 6.4 shows the AC class is built on a couple of already developed components.

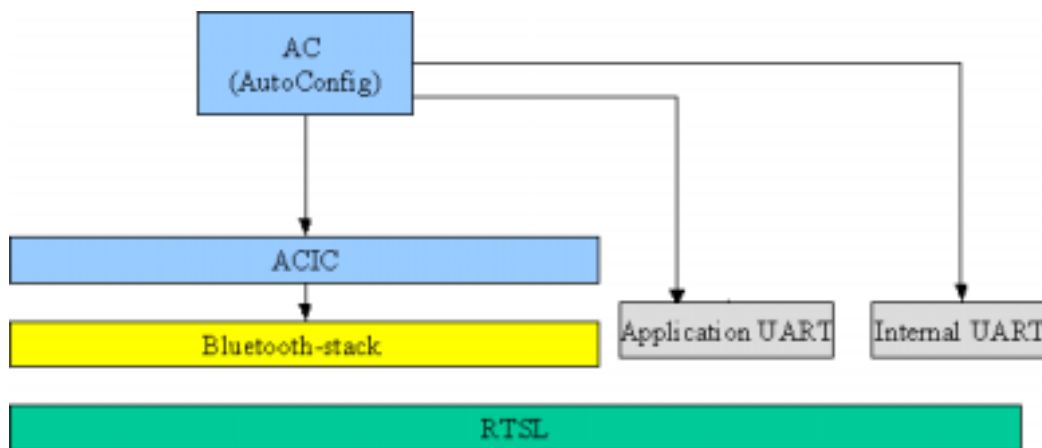


Figure 6.4: The class diagram of the modified Serial Port Adapter

The run-time support library (RTSL) consists of an operating system, software timers, and drivers such as the UART and the non-volatile data storage (NVDS). The purpose of RTSL is to make the applications on top of the connectBlue AB software independent of the hardware and operating system. An application shall be able to run on different targets as long as RTSL has been ported. The software platform is currently ported to both the PC environment and the target used in this thesis. Thus, application development can be started on a PC and then moved to the specified target. Figure 6.5 shows the architecture of RTSL.

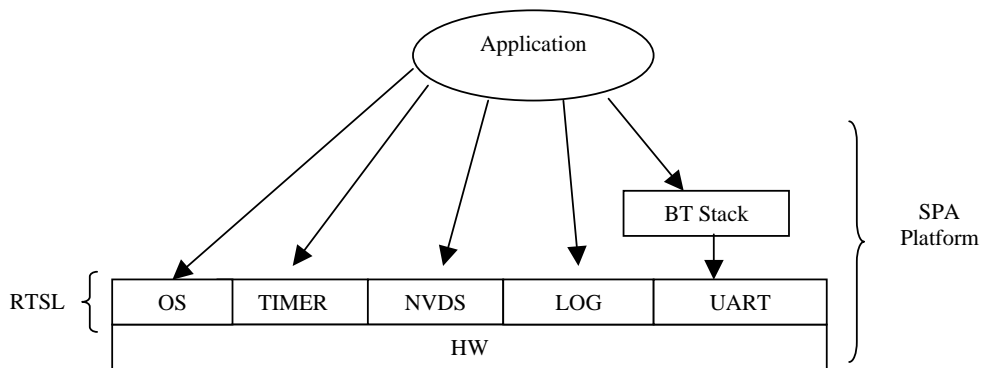


Figure 6.5: The architecture of RTSL.

The Application Controller Interface (ACI) provides Bluetooth products and applications with an interface towards a Bluetooth profile module that supports a range of Bluetooth profiles. The ACI interface has two major advantages. The first is that it forces the applications to follow the Bluetooth specification. Therefore it is much easier to qualify the product. In some cases no qualification is needed. The second is that the interface is easier to use compared to the interface of the Bluetooth stack. The profile module is called the ACI Controller and the entity that utilizes the services of the ACI Controller is called the ACI Host. The ACI Host communicates with the ACI Controller using request packets, command packets, response packets, and serial data packets. The ACI Controller communicates with the ACI Host using confirm packets, event packets, indication packets, result packets, and serial data packets. These packets are transferred between the ACI Host and the ACI Controller using a transport layer.

As illustrated in Figure 6.4, the AC class directly uses the ACI and it acts as the ACI Host. Except the advantages RTSL and ACI offer, an interface to the internal and application UART is needed in order to get all desired functionalities. Once all needed libraries are accessed, the AC class can be developed in the wanted manner.

Letting several classes share the desired functionality could also perform a single class (AC) job. Perhaps it would be easier to follow the source code in details, but it is not found necessary in this master's thesis. Showing the source code is also found unnecessary. It is owned by connectBlue AB who can be contacted for details. However, the state transition diagram, which is developed and implemented for the AC class is shown in Figure 6.6. All implementation details follow the connectBlue

recommended standard. Capital letters are used for message names while function names are written in small letters.

As shown in Figure 6.6, the modified SPA can be in one of the following 4 states: INIT, IDLE, CREATETABLE or NETWORKCONFIGURED state. The events that can happen in all of these states are gathered in the COMPOSITE state. The modified SPA is initialised in the INIT state. The configuration is performed while the SPA is in the IDLE and CREATETABLE states. When configuration is done, the SPA is in the NETWORKCONFIGURED state and data can be sent and received.

The CREATETABLE state is the state where Prim's algorithm is implemented and used to find the next device we want to connect to, i.e., the quality of links is measured. Connecting to every reachable device in turn and sending the same amount of data packets to each device is the method used for the measure. The time it takes to send the packets is the cost of the link. Shorter time means lower cost. The re-sending mechanism of not acknowledged packets makes this way to measure link quality effective and powerful.



## 7. Demonstration

A good and informative way to show the result of the solution is to make a simple demonstration. The demonstration, made for this master's thesis, is based on the modbus-protocol [16].

Modbus is an industrial serial-protocol that provides client/server communication between devices typically using the RS485-, RS422- or RS232-cable. A device acting as a modbus-master can request information from any device that is a modbus-slave. The communication in modbus is of request/response type.

The demonstration scenario is shown in Figure 7.1. This is only one scenario of many possible. Theoretically an infinite number of AC devices could be used. Because of the limited number of available SPAs, only tests that involve up to four AC devices have been performed.

Any of the present AC units can start configuration. About a minute later there is a unique way between the devices and the connection tree gets the shape shown in Figure 7.1. The temperature sensors connected to the AC units are enabled with the modbus protocol. The PC is the modbus-master in the network.

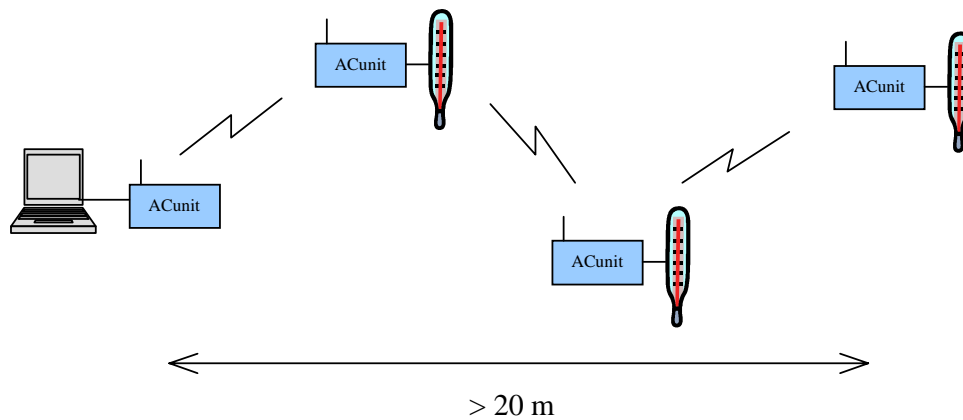


Figure 7.1: The demonstration scenario.

The application running on the PC in Figure 7.1 is developed for the purpose of this master's thesis. It is written in Microsoft Visual Basic™. When the application starts it asks which com-port to open. The com-port that is used to connect to the AC unit has to be chosen. A figure of the Graphical User Interface (GUI) is shown in Figure 7.2.

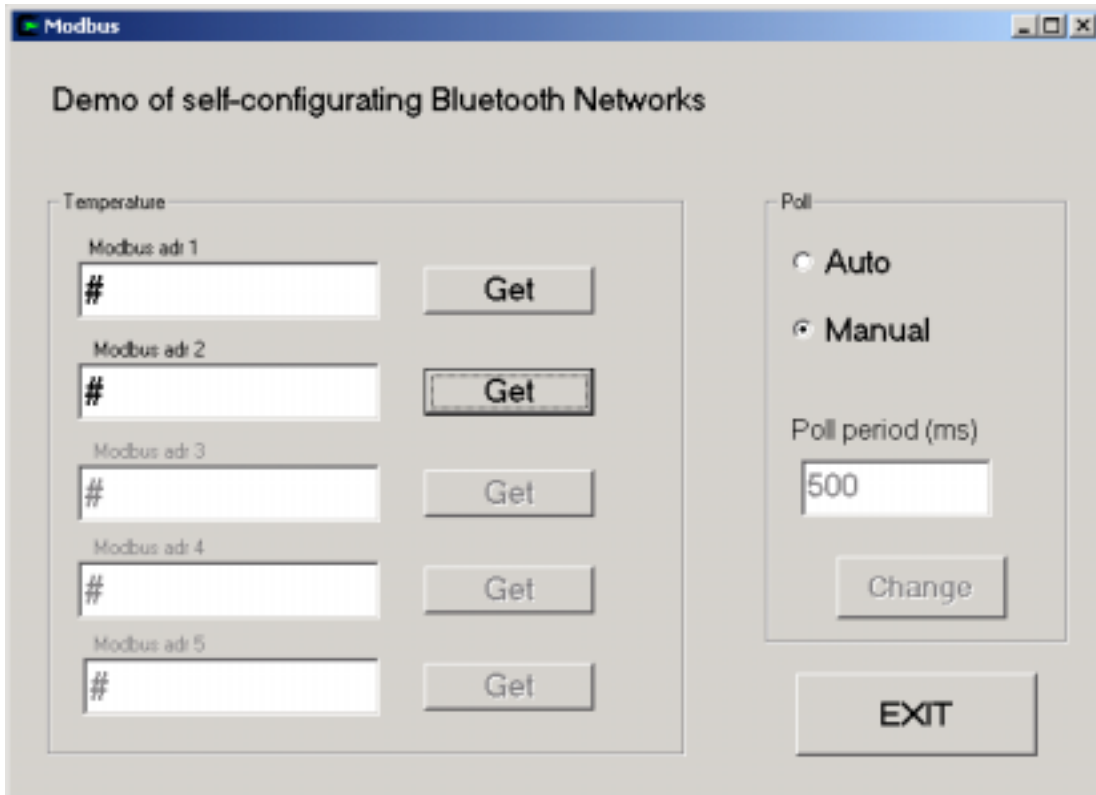


Figure 7.2: The GUI of the modbus application.

The application has two modes: automatic and manual. When in manual mode the user has to manually request the data from a modbus-slave. It happens by pressing the get button. The slave's response is shown in the corresponding textbox. When in automatic mode the application automatically polls the temperatures from all modbus-slaves. The poll period can be specified in poll period textbox.

## 8. Result

One of the main objectives of this thesis was to develop a device that supports desired scatternet features. This goal was successfully achieved by designing the AC unit. The AC unit is described in details in Chapter 6.

The algorithm developed in this thesis is built on Prim's algorithm. It prioritises high quality links in order to create as robust networks as possible. The algorithm runs independently on each AC unit and a user does not have to know anything about how the configuration is performed. Theoretically a network consisting of infinite number of devices could be created. Everything is done automatically and a led signals when the configuration is finished. The Bluetooth network created by this algorithm only involves devices we really want to participate in the network. All other devices are excluded. Once such a network is established, communication between all devices in the network is enabled. The enabled wireless communication offers the same qualities as a RS485 cable.

The time it takes to connect  $n$  number of nodes, according to this algorithm, can be approximately calculated by the following formula:

$$T(n) = t_{inquiry} \cdot n + t_{page} \cdot (n-1) + \sum_{z=1}^n k_z \cdot t_{measure} + \sum_{j=1}^n l_j \cdot t_{name\ disc} , \text{ where}$$

$n = \text{number of nodes}$

$$t_{inquiry} \approx 8,4 \text{ s}$$

$$t_{page} \approx 2,2 \text{ s}$$

$$2 \text{ s} < t_{name\ disc} < 8 \text{ s}$$

$$3 \text{ s} < t_{measure} < 6,0 \text{ s}$$

$k_z = \text{number of measurments performed by node } z$

$l_j = \text{number of devices found by node } j$

For example connecting a network that consists of three AC units within radio range takes approximately 45 seconds.

The software developed in this thesis makes the AC units capable to recover the broken links. It also supports cases when devices arbitrary arrive to an already formed network. An advantage of the developed solution is that it can easily be used for any scatternet enabled Bluetooth devices that want to create a larger Bluetooth network.

The solution made in this master's thesis was tested and a user scenario is described in the previous chapter. Implemented functionalities fulfil the desired requirements without any known limitations.

## **9. Future Work**

As already discussed in Chapter 5, there could be some user scenarios where the cheapest path from one to all other nodes is the crucial quality. Then Dijkstra's algorithm could be used in order to create an optimal scatternet-forming algorithm. Different scenarios could use different algorithms and the user should be able to choose which algorithm to use.

The current solution does not support cases where nodes arbitrary leave the network. Some kind of automatic reconfiguring of the network is desirable and could be implemented.

A list of all devices that participate in a network could be usable in order to find out if all desired devices are really involved by the network. That would be easier than checking every AC unit's led.

## **A. References**

- [1] “Specification of the Bluetooth System” v1.0 B December 1<sup>st</sup> 1999
- [2] William Stallings, “Data & Computer Communications”, sixth edition, 2000, Prentice Hall
- [3] Douglas E. Comer, “Internetworking with TCP/IP principles, protocols, and architectures”, fourth edition 2000, Prentice Hall
- [4] Brent A. Miller, Chatschik Bisdikian, “Bluetooth Revealed”, 2001, Prentice Hall
- [5] Nathan J. Muller, “Bluetooth Demystified”, 2001, The McGraw-Hill Companies
- [6] Bluetooth Special Interest Group <http://www.bluetooth.com>
- [7] <http://www-b2.is.tokushima-u.ac.jp/~ikedasuuri/dijkstra/Prim.shtml>
- [8] Jennifer Bray, Charles F Sturman, “Bluetooth connect without cables” 2000, Prentice Hall
- [9] NIST <http://www.nist.gov/dads/HTML/primsalgrthm.html>
- [10] Ching Law, Amar K. Mehta, Kai. Yeung Siu, “A new Bluetooth Scatternet Formation Protocol” Massachusetts Institute of Technology
- [11] Godfrey Tan, Allen Miu, John Guttag and Hari Balakrishnan, “Forming Scatternets from Bluetooth Personal Area Networks”
- [12] Motorola <http://ewww.motorola.com>
- [13] Nokia <http://www.nokia.com/nokia/0,1522,,00.html?orig=bluetooth/>
- [14] Ericsson <http://www.ericsson.com/bluetooth/>
- [15] Apple <http://www.apple.com/bluetooth/>
- [16] Modbus <http://www.modbus.org/default.htm>
- [17] Charles E. Perkins, Elizabeth M. Royer “Ad-hoc On-Demand Distance Vector Routing”
- [18] Jeroen Willekens “Ad Hoc Routing in Bluetooth” 2000
- [19] connectBlue AB <http://www.connectBlue.se>

## **B. Acronyms**

AC	Auto Configuring
ACIC	Application Controller Interface Controller
ACIH	Application Controller Interface Host
ACL	Asynchronous Connection-Less
ANSI	American National Standards Institute
BD_ADDR	Bluetooth Device ADDRESS
BNEP	Bluetooth Network Encapsulation Protocol
DM	Data Mode
DUN	Dial-Up Networking
FDM	Frequency Division Multiplexing
GAP	Generic Access Profile
GUI	Graphical User Interface
HCI	Host Controller Interface
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
LAN	Local Area Network
L2CAP	Logical Link Control and Adaptation Protocol
LC	Link Controller
LM	Link Manager
LMP	Link Manager Protocol
MAC	Media Access Control
OBEX	OBject Exchange protocol
OSI	Open Systems Interconnection
PAN	Personal Area Network
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PPP	Point To Point Protocol
RFCOMM	Protocol for RS-232 serial cable emulation
RTSL	RunTime Support Library
SCO	Synchronous Connection Oriented
SDP	Service Discovery Protocol
SIG	Special Interest Group
SPA	Serial Port Adapter
SPP	Serial Port Profile
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TCS-BIN	Telephony Control protocol Specification – BINary
UART	Universal Asynchronous Receiver Transmitter

### C. The Bluetooth Stack

The Bluetooth protocol stack is a layered model of different working protocols that work together [1]. Each protocol uses the services of the protocol below in order to solve the requests of the protocol above. The Bluetooth stack is shown in Figure C.1 and a comparison with the OSI reference model is shown in Figure C.2.

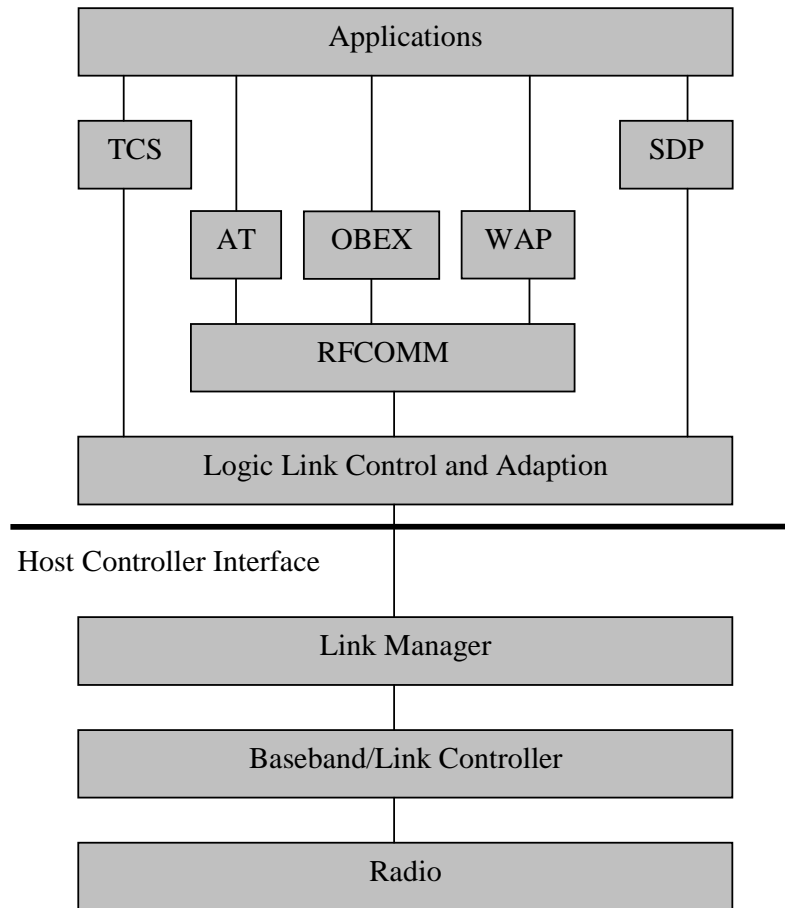


Figure C.1: Bluetooth Protocol Stack.

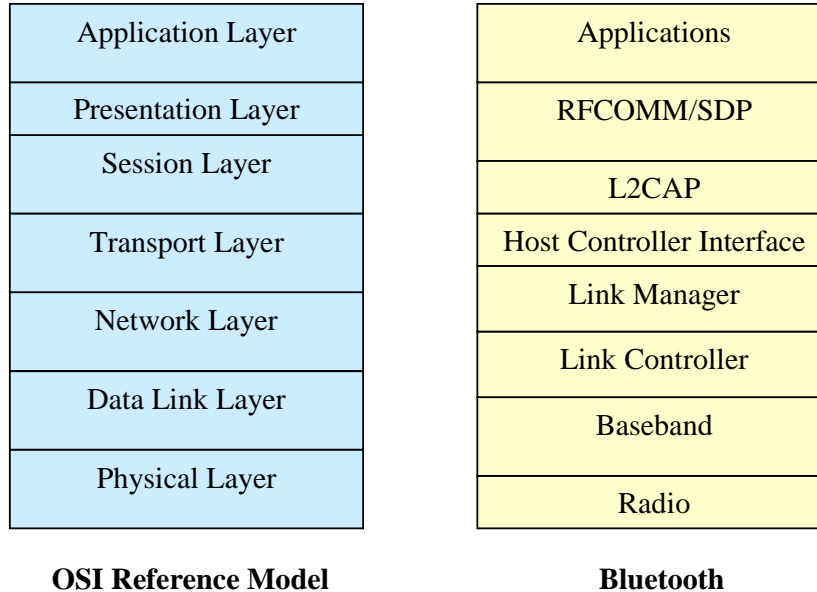


Figure C.2: Comparison between OSI reference model and the Bluetooth stack.

Below is a brief explanation of Bluetooth stack layers. For detailed information the Bluetooth Specification v1.1 is recommended to read.

### C.1 Radio

Bluetooth radio modules use 2.4 GHz Industrial Scientific and Medicine (ISM) band [8]. In order to be robust against disturbances Bluetooth radio uses a technique called spread-spectrum frequency hopping. In this technique, a device will use 79 individual, randomly chosen frequencies within a designated range, changing from one to another on a regular basis. In the case of Bluetooth, the transmitters change frequencies 1,600 times every second, meaning that more devices can make full use of the radio spectrum. This makes it unlikely that two transmitters will be on the same frequency at the same time. Each device is classified into 3 power classes, Power Class 1, 2 & 3.

- Power Class 1: is designed for long-range (~100 m) devices, with a max output power of 100 mW (20 dBm).
- Power Class 2: for ordinary range devices (~10-15 m) devices, with a max output power of 2.5 mW (4 dBm).
- Power Class 3: for short-range devices (~10 m) devices, with a max output power of 1 mW (0 dBm).

## C.2 Baseband

The Baseband layer lies on top of the Bluetooth radio layer in the Bluetooth stack. It is responsible for channel coding/decoding, low-level timing control and management of the link within the domain of a single data packet. The Link Controller works with the link manager for carrying out link level routines like link connection and power control. All Bluetooth devices have a unique 48-bit Bluetooth Address (BD\_ADDR).

The Baseband handles two types of links: Synchronous Connection-Oriented (SCO) and Asynchronous Connection-Less (ACL) link.

The SCO link is a symmetric point-to-point link between a master and a single slave in the piconet. The master maintains the SCO link by using reserved slots at regular intervals. Thus, the SCO link provides a circuit-switched connection and mainly carries voice information. The master can support up to three simultaneous SCO links. SCO packets are never retransmitted and are used for 64 kB/s speech transmission.

The ACL link is a point-to-multipoint link between the master and all the slaves participating on the piconet. In the slots not reserved for the SCO links, the master can establish an ACL link on a per-slot basis to any slave, including the slave already engaged in an SCO link. Thus the ACL link provides a packet-switched connection where data is exchanged sporadically. Only one ACL link can exist between any two devices. For most ACL packets, packet retransmission is applied.

A connected Bluetooth device can be in any of the following four modes: Active, Hold, Sniff and Park mode. When not connected the device is in standby mode.

**Active Mode:** In the active mode, the Bluetooth unit actively participates on the channel. Active slaves listen in the master-to-slave slots for packets that enable them to remain synchronized with the master and inform them when they can transmit packets back to the master. If an active slave is not addressed, it may sleep until the next new master transmission.

**Park Mode:** In the park mode, a device is still synchronized to the piconet but does not participate in the traffic. Parked devices occasionally listen to the traffic of the master to re-synchronize and check on broadcast messages. Parked mode is the least responsive of the connected modes, since the slave must become an active member of the piconet before resuming communications. Park mode has the lowest power consumption.

**Sniff Mode:** Sniff mode is one method for reducing power consumption. A slave becomes active periodically, which makes it less responsive but reduces its power consumption. The sniff interval is programmable and is set by the application.

**Hold Mode:** In hold mode a slave stops listening for packets entirely for a specified time interval. During the hold time the units do not need to listen for packets from the master. After the hold period the slave resumes listening for packets from the master.

### **C.3 Link Manager**

The Link Manager is responsible for maintaining a link once it has been set up. It discovers other remote LM's and communicates with them via the Link Manager Protocol (LMP). To perform its service provider role, the LM uses the services of the underlying Link Controller (LC).

The Link Manager Protocol essentially consists of a number of PDU (protocol Data Units), which are sent from one device to another, determined by the AM\_ADDR in the packet header.

### **C.4 Host Controller Interface**

The radio, baseband and link manager may be packaged together into a Bluetooth module. The module is then attached to a host device, via RS232-port or UART, enabling the device with Bluetooth wireless communication. This means that the host must contain L2CAP and any other appropriate higher layers of the protocol stack.

The HCI provides a uniform command interface of accessing the Bluetooth hardware. The HCI Link commands provide the host with the ability to control the link layer connections to other Bluetooth devices.

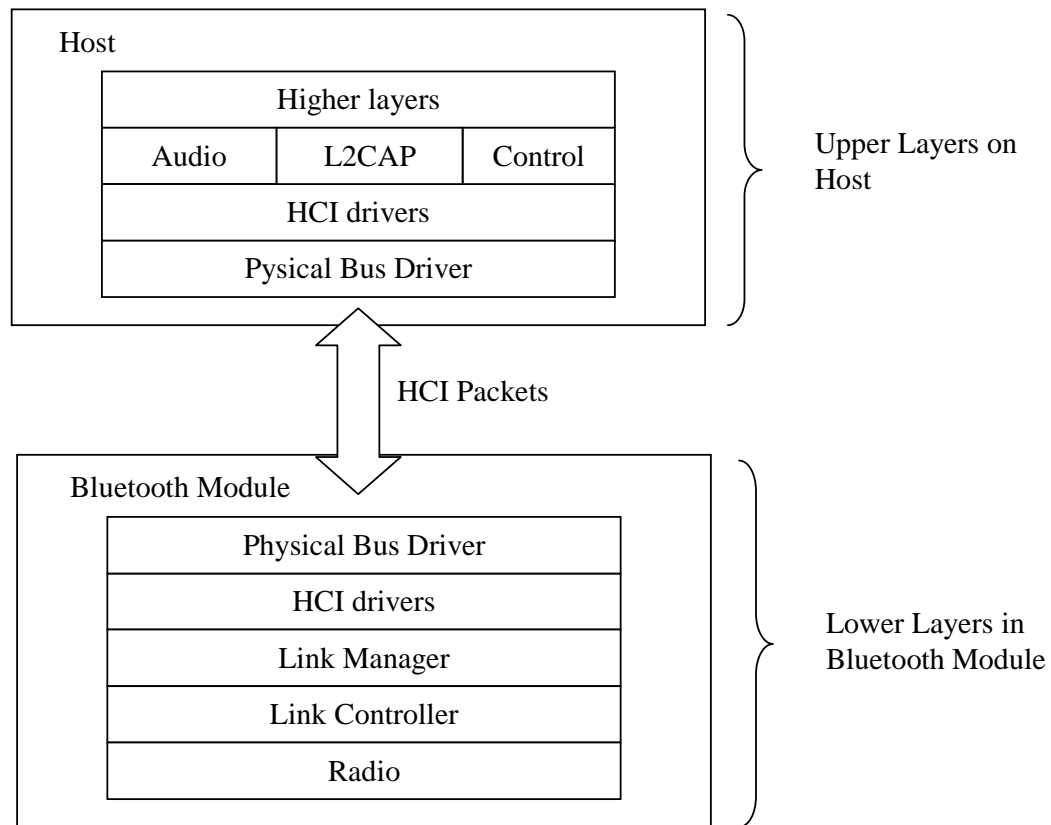


Figure C.3: Location of the HCI in the Bluetooth protocol stack.

### C.5 Logical Link Control and Adaptation Protocol

The Logical Link Control and Adaptation Layer Protocol (L2CAP) is layered over the Baseband Protocol and resides in the data link layer. L2CAP provides connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group management. L2CAP permits higher-level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

L2CAP must support protocol multiplexing because the Baseband Protocol does not support any type-field identifying the higher layer protocol being multiplexed above it. L2CAP must be able to distinguish between upper layer protocols such as the Service Discovery Protocol, RFCOMM and Telephony Control.

The Segmentation and Reassembly functionality is absolutely necessary to support protocols using packets larger than those supported by the Baseband. The maximum transmission unit associated with the largest Baseband payload (341 bytes for DH5 packets) limits the efficient use of bandwidth for higher layer protocols that are designed to use larger packets. Large L2CAP packets must be segmented into multiple smaller Baseband packets prior to their transmission over the air. Similarly,

multiple received Baseband packets may be reassembled into a single larger L2CAP packet following a simple integrity check.

## **C.6 RFCOMM Protocol**

RFCOMM is a simple transport protocol, which provides emulation of RS232 serial ports over the L2CAP protocol. The specification allows for up to 60 multiplexed logical serial links over a single RFCOMM connection. But the actual number of connections that can be used simultaneously in a BT device is implementation-specific. For the purposes of RFCOMM, a complete communication path involves two applications running on different devices with a communication segment between them.

Basically the RFCOMM supports two device types:

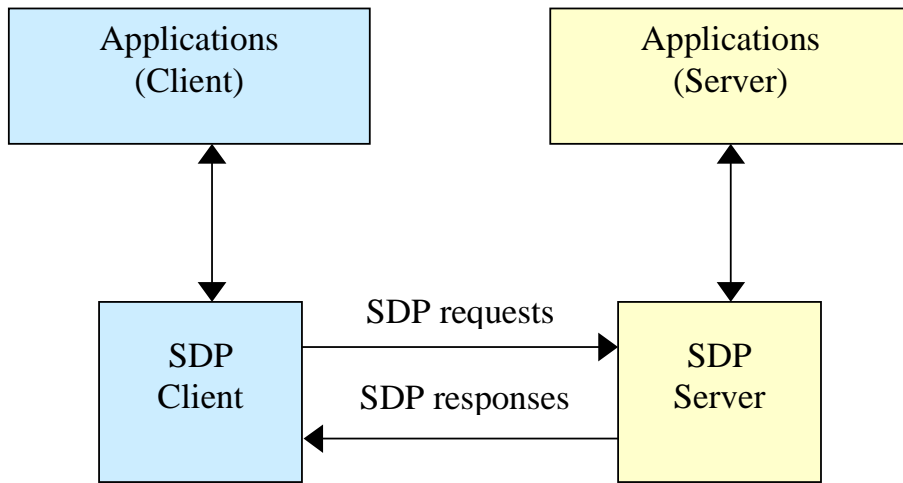
- Type 1, Internal emulated port (or equivalent).
- Type 2, Intermediate device with physical serial port.

Though RFCOMM does not make a distinction between these two device types in the protocol, accommodating both types of devices impacts the RFCOMM protocol. The information transferred between two RFCOMM entities has been defined to support both type 1 and type 2 devices. Type 2 devices only need some information while some information is intended to be used by both types. In the protocol, no distinction is made between type 1 and type 2. Since the device is not aware of the type of the other device in the communication path. This means that each device must pass on all available information specified by the protocol.

## **C.7 Service Discovery Protocol**

In traditional networks, such as LANs, the network administrator statically configures services. In Bluetooth, the end user performs the configuration that is necessary to participate in the network. A specific Service Discovery Protocol (SDP) is needed in the Bluetooth environment, as the set of services that are available changes dynamically based on the RF proximity of devices in motion. The Service Discovery Protocol provides means for applications to discover which services are available on a remote Bluetooth device and to determine the characteristics of those available services.

SDP uses a request/response model where each transaction consists of one request protocol data unit (PDU) and one response PDU. In the case where SDP is used with the Bluetooth L2CAP transport protocol, only one SDP request PDU per connection to a given SDP server may be outstanding at a given instant. In other words, a client must receive a response to each request before issuing another request on the same L2CAP connection. Limiting SDP to sending one unacknowledged request PDU provides a simple form of flow control.



*Figure C.4: The SDP Protocol.*

## D. Bluetooth profiles

The Bluetooth profiles have been developed in order to describe how implementations of user models are to be accomplished. A profile can be described as a vertical slice through the protocol stack. It defines options and parameter ranges in each protocol that is mandatory for the profile. The profile concept is used to guarantee interoperability between devices of different manufactures [6].

The Bluetooth profile structure and the dependencies of the profiles are depicted below. A profile is dependent upon another profile if it re-uses parts of that profile. The dependencies are illustrated in the Figure D.1: a profile has dependencies on the profile(s) in which it is contained. For example, the File transfer profile is dependent on Generic Object Exchange, Serial Port, and Generic Access profiles.

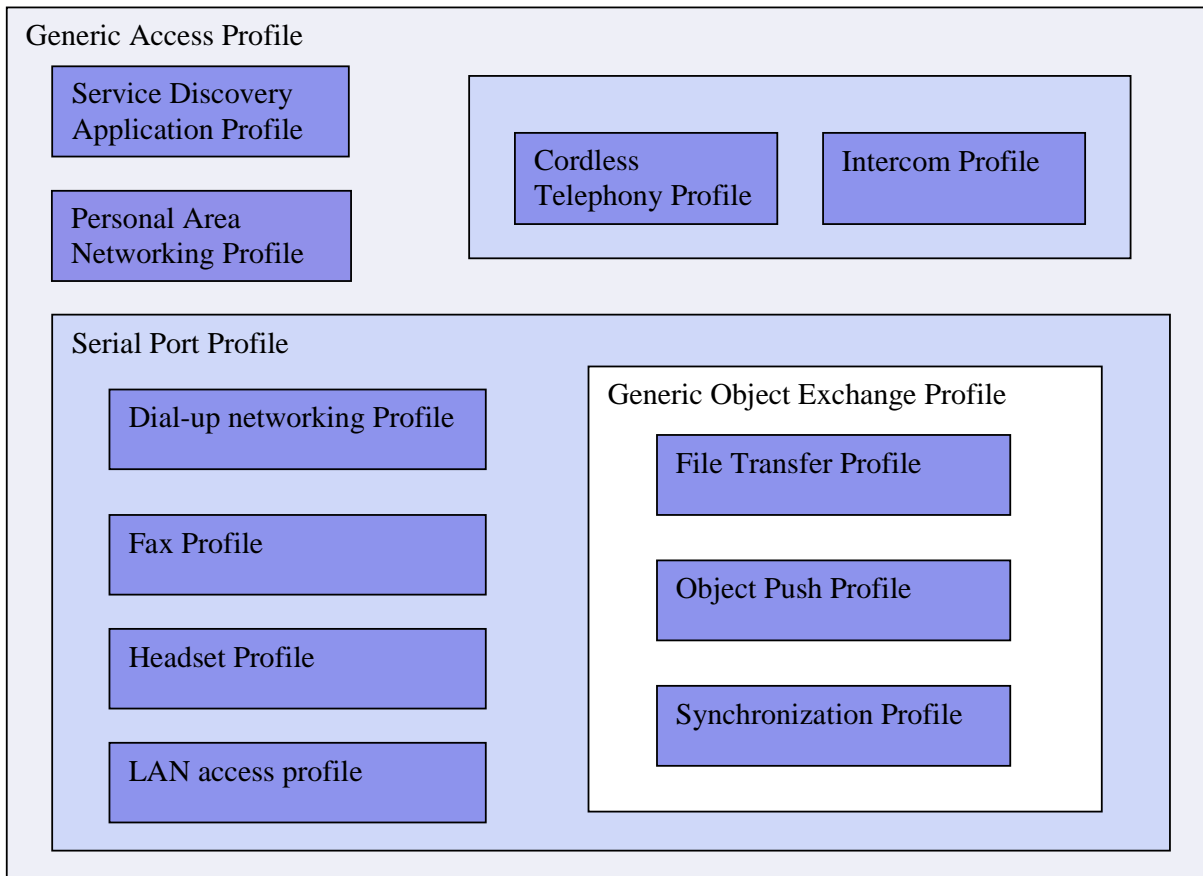


Figure D.1: Bluetooth Profiles and there dependencies.

The current Bluetooth Profiles specification (version 1.1) calls for support the following profiles:

- **General Access**, the basis for all profiles. It describes the fundamental operations necessary for two Bluetooth devices to establish a connection, including the discovery of devices.
- **Service Discovery**, discover services available in other Bluetooth devices.
- **Serial Port**, emulated serial port for applications that traditionally use a wired serial port interface.
- **Headset**, defines both the transmitting and receiving voice data over a Bluetooth link.
- **Dial-up Networking**, defined primarily as the link between a cell phone & a computer. The profile is defined so the computer can either initiate the data call or receive the data call depending on user setup.
- **Cordless Telephony**, ability for a cell phone to operate as a cordless phone when in proximity to its wired telephone service base station.
- **Intercom**, defines the intercom function in cell/mobile phones using Bluetooth as the radio link.
- **Fax**, cell phone or modem may be used as a fax interface to a computer for both sending and receiving faxes.
- **LAN Access**, setting up a personal area network using Point-to-Point Protocol (PPP).
- **Generic Object Exchange (OBEX)**, defines procedures used by the files transfer, object push, and synchronization profiles
- **Object Push**, ability to push or pull a business card or an appointment
- **File Transfer**, browse a file system on, create or delete files / folders on, or transfer files from/to another Bluetooth device
- **Synchronization**, exchange of personal information such as calendar & phonebook. Also the use of a mobile phone or computer to automatically start synchronization when in range
- **Personal Area Networking (PAN)**, provides network capabilities to Bluetooth devices and lays out the rules for carrying IP-traffic across Bluetooth connections.

The most important profiles for this master's thesis are explained below.

### D.1 Generic Access Profile

The General Access Profile (GAP) is the basis for all the other Bluetooth profiles. It describes the fundamental operations necessary for two Bluetooth devices to establish a connection, including the discovery of devices. All Bluetooth devices must conform to at least the GAP to ensure basic interoperability between devices.

The Bluetooth user should in principle be able to connect a Bluetooth device to any other Bluetooth device. Even if the two connected devices does not share any common application, it should be possible for the user to find this out using basic Bluetooth capabilities. When the two devices do share the same application but are

from different manufacturers, the ability to connect them should not be blocked just because manufacturers choose to call basic Bluetooth capabilities by different names on the user interface.

The GAP Profile modes are:

- Discoverability, when a device is in discoverable mode, it shall periodically enter the Inquiry scan mode so other Bluetooth devices can find it.
- Connectability, when a device is in connectable mode, it shall periodically enter the page scan mode so other Bluetooth devices can connect to it.
- Pairability, allows a Bluetooth device to be authenticated by another Bluetooth device.

## D.2 Serial Port Profile

The Serial Port Profile SPP is a transport protocol profile that defines the fundamental operations necessary to establish RFCOMM connection between two peer devices. The serial cable emulation can then be used for tasks such as data transfer and printing. The requirements are expressed in terms of services provided to applications, and by defining the features and procedures that are required for interoperability between Bluetooth devices.

Essentially, the Serial Port Profile defines the protocols and procedures that shall be used by devices using Bluetooth for RS232 serial cable emulation. The scenario covered by this profile deals with applications using Bluetooth as a cable replacement, through a virtual serial port abstraction.

## D.3 Dial-up Networking Profile

The Dial Up Network Profile (DUN) provides a dial up connection. This allows for example a Lap Top to access a telephone network using the services of a communicating device such as a cellular phone or a modem.

Two main scenarios are implemented: the Usage of a cellular phone by a computer as a wireless modem or using it for connecting to a dial-up internet access.



*Figure D.2: A type of Dial Up Network*

## D.4 LAN Access Profile

The LAN access profile allows a Bluetooth enabled device to access a fixed network via a Bluetooth link to a LAN Access Point (LAP). The Bluetooth device would be connected to the LAN as an ordinary LAN-station, and the transmission would use the PPP (Point-to-Point Protocol). But the LAN Access Profile for Bluetooth devices consists of 2 parts. It also shows how the same PPP mechanisms are used to form a network consisting of two Bluetooth-enabled devices.

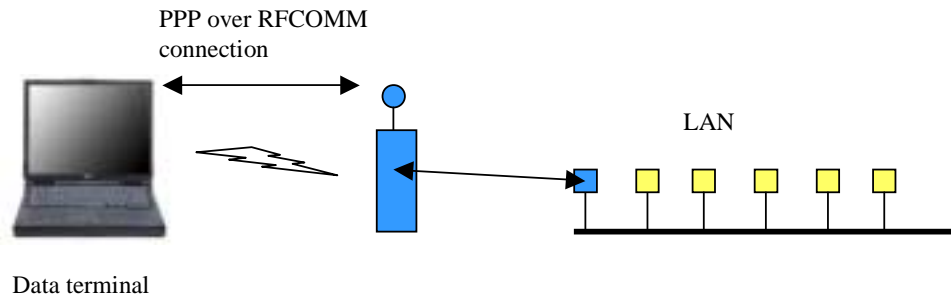


Figure D.3: The LAN Access Profile.

## D.5 Personal Area Networking Profile

The PAN profile lays out the rules for carrying IP traffic across Bluetooth connections. It relies upon the Generic Access Profile, but does not make use of any other profile. The PAN profile will use the still not final Bluetooth Network Encapsulation Protocol (BNEP), to provide network capabilities. In today's version of PAN, it focuses on a simple networking scenario consisting of a single Bluetooth piconet. The PAN profile does not support automatic network formation.

A Bluetooth device that supports the PAN profile can have one of the following three roles:

**Network Access Point (NAP):** A Bluetooth device that supports the NAP-role is capable of bridging IP-packets between a piconet and a LAN. Hence it is possible to use it as an access point to LAN.

**Group Ad-hoc Network:** A Bluetooth device that supports the GN-role is capable of forwarding packets between two PAN users.

**PAN user (PANU):** A Bluetooth device that supports the PANU-role is capable of connecting to Bluetooth devices supporting the NAP- or GN-services. A PANU can also connect directly to a PANU for IP-traffic.