AerotechTelub

Simulation of Bluetooth Network

Lennart Lagerstedt

Stockholm, 2003

Master of Science Thesis Project

The Department of Microelectronics and Information Technology,

Royal Institute of Technology (KTH)

Simulation of Bluetooth Network

Abstract

Bluetooth – a new technology that allows different types of devices to form temporary networks over a wireless radio interface, has aroused an enormous interest among thousands of companies around the world. Cable replacement with help of Bluetooth's radio interface to supply data and voice links can change and facilitate the everyday work.

The purpose of this Master thesis is to describe a model of a Bluetooth network which is implemented in OPNET Modeler, a modern simulation tool for communication networks. Interesting performance parameters like throughput, delay and resource allocation are some of the parameters, which the user can study after a simulation.

Results from two different simulation scenarios, one with only file transfer and one with only voice communication, are illustrated with the above mentioned parameters.

Acknowledgements

I will give many thanks to my supervisors Kennert Andersson and Jonas Nilsson at AerotechTelub who have helped me with the work in many different ways. Especially Kennert's knowledge of the radio communication area and of the computer network area has helped me much.

Henrik Vaks and Anumeet Singh are two other persons who I want to thank for their support during the implementation phase.

I will also thank my examiner Thomas Sjöland, Director of studies, and supervisor Prof. Seif Haradi both from KTH.

Table of contents

1 Introduction

1.1 Background

Bluetooth is a quickly growing technology within the area of wireless communication. The possibility to integrate small and cheap chips into different types of devices to let them be able to communicate with each other has attracted about 1400 companies around the world to join the Bluetooth Special Interest Group (SIG). The Bluetooth technology is an open specification and this means that any company can get a free copy of the Bluetooth Specification if they are interested in the new standard.

The biggest advantage with Bluetooth is that people do not need to use cables any more. The wireless connection that Bluetooth offers can for example be used to replace cables between computers and printers or computers and mobile phones. Bluetooth can be used to handle both data and voice links.

Low complexity, low power and low cost are some words that characterise the features, which Bluetooth offers.

The first products are expected on the market some time during the second half of this year.

1.2 The purpose of this Master thesis

The purpose of this Master thesis is with help of the Bluetooth Specification [1] to build up a model of a Bluetooth piconet. The model will then be implemented into OPNET Modeler, which is a program for simulation of communication networks.

Some interesting things that the implementation shall be able to illustrate are:

- Transmission rates
- Connection delays
- Packet losses
- Power consumption
- Etc

1.3 Sequence of work

The following steps have been used for the realisation of the Master thesis:

- Collection of information about Bluetooth
- Study of the Bluetooth technology
- Design of the model
- Implementation of the model
- Simulations for verification of the model
- Report writing to collect results and experience

1.4 The disposition of this report

Chapter 2 gives an overview of the protocol stack, which Bluetooth uses. Some examples of what the Bluetooth technology can be used for is also given.

Chapter 3 introduces characteristics of the radio part of Bluetooth.

Chapter 4 describes Bluetooth's Baseband very carefully. The reason to this is that most of the model's functionality is based on the baseband layer.

Chapter 5 is an introduction to the profiles that are defined for Bluetooth.

Chapter 6 describes how the implemented model is intended to work.

Chapter 7 describes a bit about how the implementation can be done with OPNET Modeler.

Chapter 8 gives a description of the implementation.

Chapter 9 describes how simulations can be done with help of the different attributes of the present devices.

Chapter 10 shows a couple of simulation scenarios and corresponding results.

Chapter 11 gives examples of how the model can be expanded.

Chapter 12 summarises some results and experiences.

The report ends with abbreviations, an appendix and references.

2 Overview

2.1 Protocol stack overview

Figure 2.1 below shows how Bluetooth's most fundamental protocol layers are related to each other.

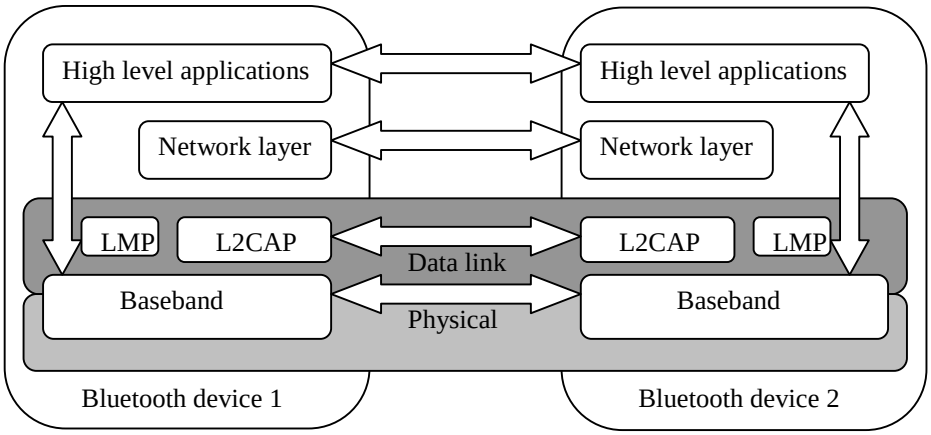

Figure 2.1: Bluetooth's protocol layers.

The following subsections describe the functionality of the protocols, which are used in the Baseband layer, L2CAP layer and LMP layer.

Notice that the protocol stack consists of further protocol layers on the top of the stack but these are not mentioned here.

2.1.1 Baseband Protocol

The Baseband Protocol is used to control the signal processing part of the Bluetooth hardware, the Bluetooth link controller. The Baseband belongs to the physical layer of the protocol stack.

The Baseband Protocol defines two different types of physical links:

- SCO link (Synchronous Connection-Oriented, used for voice links)
- ACL link (Asynchronous Connection-Less, used for data links and control)

These two link types define how baseband packets shall be handled. Before a packet from a higher layer can be transmitted, it will be segmented (if necessary) into smaller baseband packets.
The Baseband Protocol also takes care about different access procedures like inquiry and page. The inquiry procedure makes it possible for a Bluetooth device to locate surrounding devices. The page procedure can then be used by the (becoming) master if some of the earlier inquired devices shall join the network.

Some other features that the baseband controls are:

- Flow control

- Synchronisation
- Transmit and receive timing
- Error correction
- Security
- Etc

See also section 4, Baseband Specification, for more detailed information.

2.1.2 L2CAP – Logical Link Control and Adaptation Layer Protocol

The L2CAP layer is located over the Baseband Protocol and belongs to the data link layer. L2CAP provides both connection-oriented and connection-less data services to upper layer protocols. Some functionality requirements that the L2CAP layer must be able to handle are:

- Protocol multiplexing

As the Baseband Protocol does not provide protocol multiplexing of higher layers the L2CAP Protocol must take care of that.

- Segmentation and reassembly

If necessary, large L2CAP packets must be segmented into smaller packet sizes before the Baseband can transmit them over the air interface. On the receiver side the packets must be able to be reassembled again before they can be sent to higher layers.

- Quality of Service

The L2CAP layer must be able to exchange information regarding the expected quality of service. Quality of Service parameters can for example be transmission rate or transmission latency.

2.1.3 LMP – Link Manager Protocol

The Link Manager Protocol is used for link set-up, security and control. The link messages, which will be transferred between two Bluetooth devices, are carried in the payload field of the packet. These link messages have higher priority than ordinary user data and shall therefore not be delayed by the present traffic.

Link messages are always sent in single slot baseband packets.

2.2 What can Bluetooth be used for?

The following sections describe some practical examples (user models) to show what Bluetooth can be used for. The best benefit with Bluetooth is that the users do not have to get tangled into cables.

2.2.1 Three-in-One phone

At home, your mobile phone functions as a cordless phone, and you are just paying the normal line charges. When you are leaving your home, the mobile phone is working like a usual mobile phone. Then when entering the office, your mobile phone automatically connects to the company's local telephone net. You will not be charged for the company's phone calls.

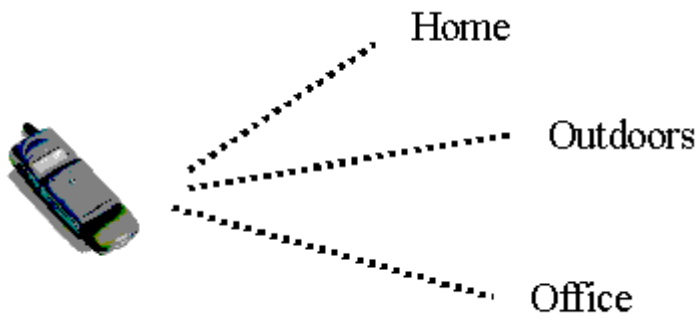

Figure 2.2: Three-in-one phone.

2.2.2 The cordless office

Cordless connection of your desktop, laptop, printer and scanner makes your life easier. Now you do not have to struggle with all those cables.

A cordless connection of your mouse and keyboard to your PC or a cordless connection to a LAN is also possible.

2.2.3 Synchronisation

If you have a desktop, a portable computer, a mobile phone and maybe a Personal Digital Assistant (PDA) it is not easy to have the information updated between these devices. An example is your electronic calendar. Bluetooth can for example offer automatically updates (synchronisations) between your desktop and PDA, when you are entering your office.

Different types of automatically backups between your PDA and mobile phone can also be useful.

2.2.4 File Transfer

If you have a portable computer and want to send a file to your partner, you can get a cordless connection to the LAN. The actual file is then transmitted from your computer to the actual destination via the LAN.

3 Radio specifications

3.1 Introduction

Bluetooth radio transmitters use the air as a transport medium to communicate with each other. The basic idea is that a device shall have the possibility to set up cordless radio LANs with other devices in its surrounding. The range of the transmitting antenna will normally be up to 10 meters but if necessary a power amplifier can increase the distance up to 100 meters.

3.2 Frequency usage

A transmitting Bluetooth device is operating in the 2.4 GHz ISM (Industrial Scientific Medicine) band. The ISM band is globally available around the world but some countries have restrictions and cannot therefore use the same bandwidth range. The total capacity of the channel is divided into several frequency bands and a frequency-hopping algorithm manages access to the channel. In Table 3.1 below are the regulatory ranges and different types of defined channels illustrated. The channel spacing is 1 MHz.

Geography	***Regulatory Range***	***Channels***
USA	2.400-2.4835 GHz	f=2402+k MHz, k=0,…,78
Europe[1]	2.400-2.4835 GHz	f=2402+k MHz, k=0,…,78
Spain	2.445-2.475 GHz	f=2449+k MHz, k=0,…,22
France	2.4465-2.4835 GHz	f=2454+k MHz, k=0,…,22
Japan	2.471-2.497 GHz	f=2473+k MHz, k=0, …, 22

Table 3.1: Frequency bands depending on location.

3.3 Transmitter characteristics

The radio transmitter can work with different power levels. These are represented with three power classes, which set up limits for maximum and minimum output

[1] France and Spain are not included.

power. For example, power class 1 has a maximum output power of 100 mW (20 dBm) and can be used with an optional connector. The most ordinary operating class so far is however power class 3 and it has a maximum output power of 1 mW (0 dBm).

Bluetooth devices have the opportunity to control the power level during a present transmission. A dynamically output power level could be used to optimise the power consumption or reduce the interference level. For power class 1, the power control is mandatory for transmitting power over 0 dBm. Otherwise, the power control for the other classes is optional.

The modulation method is GFSK (Gaussian Frequency Shift Keying).

3.4 Receiver characteristics

Bluetooth devices have some interference performance to live up to. The most important interference demand is the Co-channel interference, i.e. interference from signals with the same frequency. The ratio between the wanted signal and the interfering signal must be at least 11 dB; otherwise the receiver will not be able to detect the carried information.

4 Baseband specifications

4.1 Introduction

The transceiver works with a frequency hop sequence, in the unlicensed ISM band at 2.4 GHz, to try to minimize the interference and fading. The channel spacing is 1 MHz and starts at 2.402 GHz.. Figure 4.1 below exemplifies how the frequency (different bands) is changed during the transmission.

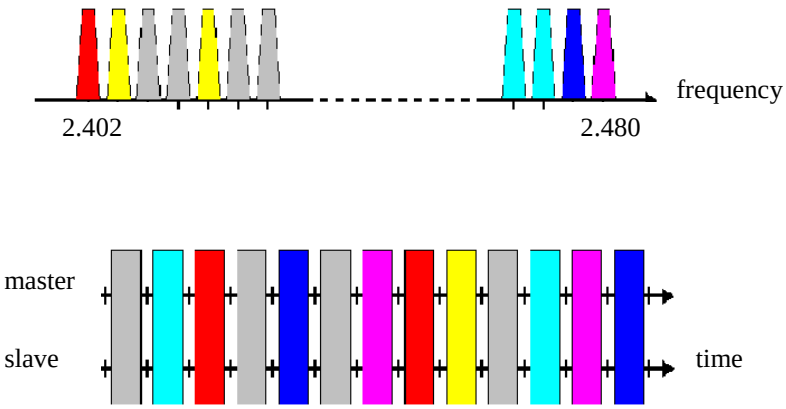

Figure 4.1: Frequency hopping.

Bluetooth works with packet transmission over a slotted channel, which has a symbol rate of 1 Mbit/s. The nominal slot length is 625 µs and it is the maximum length as a normal packet may cover, but packets can also be extended to cover three or five slots. To handle full duplex transmission a Time-Division Duplex (TDD) scheme is used.

Bluetooth supports both asynchronous and synchronous channels. Asynchronous channels can be used to transmit ordinary data in one single slot. In the synchronous case, several slots are reserved and that makes it possible to support voice channels. Up to three simultaneous synchronous voice channels can be handled at the same time and each voice channel offers 64 kb/s in both directions. A Bluetooth device has also an option to simultaneously transmit both asynchronous data and synchronous voice.

Bluetooth devices communicate with each other with a point-to-point connection or with a point-to-multipoint connection. In the latter case the channel is shared among several devices.

4.2 Master - Slave relationship

At least two devices must be represented if a Bluetooth network shall be able to exist. Two or more devices that share the same frequency hop sequence form a *piconet*. One device within the piconet acts as the master and the other device(s) act as slave(s).

The responsibility of the master is to control and manage the traffic within the piconet. This means that the master can force slaves to occupy different states depending on how the slave shall act.

4.3 Piconet

As mentioned before, two or more devices that are sharing the same frequency hop sequence form a *piconet*. Each piconet can only have one master and a maximum of seven active slaves. Nothing prevents more than seven slaves to belong to the actual piconet, but in this case the remaining slaves cannot be active at the same time. A slave can for example be in a so-called parked state. Figure 4.1 below shows two different piconets, one with only one slave and the other with three slaves.

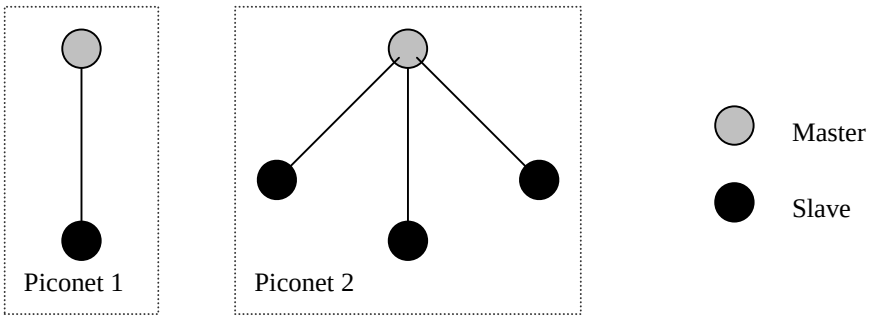

Figure 4.1: Two piconets with different number of slaves.

4.4 Scatternet

If there are two or more piconets that overlap each other, we have a *scatternet*. Overlapping piconets means that a slave belongs to more than one piconet. For example, a master in one piconet can at the same time be a slave in another piconet.

To enable a slave to participate simultaneously in different piconets, a time-division multiplex scheme must be used. The thought is not that different piconets

shall be time or frequency synchronised, instead, each piconet has its own hopping sequence. The actual device must use different hopping channels. Figure 4.2 below shows a scatternet.

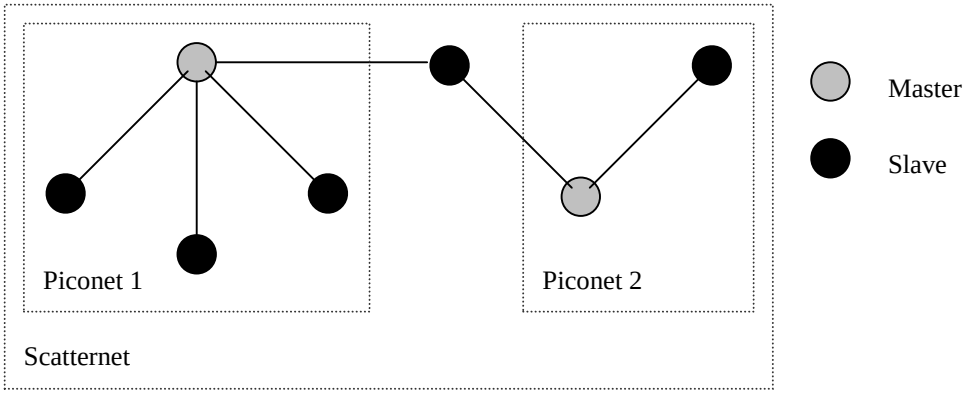

Figure 4.2: A scatternet. The slave in the centre of the figure participates with two different piconets.

Every packet sent on the piconet channel has a unique *channel access code* (CAC). If multiple piconets cover the same area, a slave can participate in two or more overlaying piconets by applying time multiplexing. The packets, which belong to two or more channels, are being identified with their different channel access codes.

4.5 Master-slave switch

The unit that creates a piconet will become the master of the piconet. If it is necessary for another slave in the piconet to become master, a *master-slave switch* can be done. This switch will result in a redefinition of the actual piconet, as the new piconet parameters are derived from the device address and clock of the new master. This procedure is also called a *piconet switch.*

As a consequence of a piconet switch, other possibly involved slaves must also adapt to the new piconet parameters.

4.6 Channel

Before devices will be able to communicate with each other within a piconet a channel must be established. The channel is represented by a pseudo-random hopping sequence, which uses 79 radio frequency (RF) channels to alternate between. The number of available RF channel depends on the location; see Table 3.1 for further information. The channel is divided into time slots where each slot corresponds to different hop frequencies. After a time slot has been used with a

certain frequency, a new frequency among the different RF channels is calculated for the next time slot. The new frequency is always calculated on the basis of the master's clock and device address.
The nominal hop rate is 1600 hops/s.

In some cases the transmitting frequency is not changed. The exception is when a device shall send information over several time slots, i.e. when asynchronous channels are used. In this case the RF hop frequency for the entire packet is derived from the master's clock value for the first time slot of the packet. All of the devices that are participating in the communicating network still follow the pseudo-random hopping sequence and it is just the present transmitting device and receiving device, which hold the actual frequency.

4.7 Transmission between master and slave

The time slots are numbered with a range from 0 to 2^{27} - 1, with a time cycle length of 2^{27}. Each time slot has a length of 625 µs in which a single packet can be transmitted. The master and for the moment present slave take turns transmitting packet over the channel. To handle that behaviour a Time-Division Duplex (TDD) scheme is used. The master always starts its transmission in even-numbered time slots, while the slave uses odd-numbered time slots. Packets with sizes up to five time slots may be transmitted.

Figure 4.3 below shows how the alternating transmission works with one slot usage.

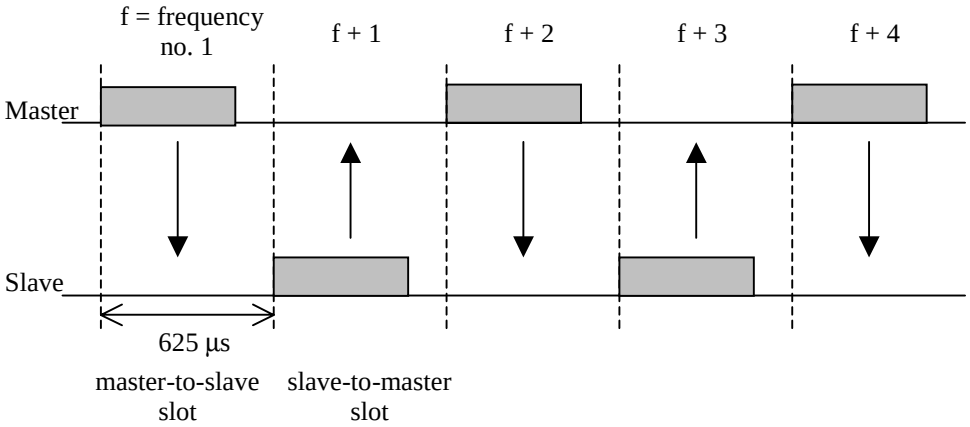

Figure 4.3: Alternating transmission between master and slave with different frequencies.

If the transmission contains packets with different sizes, multi-slot packets must be used. The entire multi-slot packet shall be transmitted with the same hop

frequency as the applied hop frequency of the first time slot. Figure 4.4 below shows two examples of transmission of multi-slot packets.

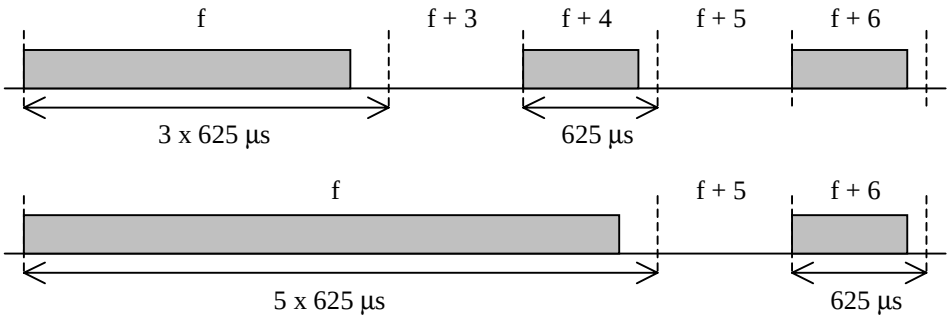

Figure 4.4: Transmission of multi-slot packets.

4.8 Links

There are two different types of links, which can be established between the master and the slave(s). These are as follows:

- Asynchronous Connection-Less (ACL) link
- Synchronous Connection-Oriented (SCO) link

The ACL link is a point-to-multipoint link between the master and all participating slaves. The SCO link is just a point-to-point link between the master and a single slave. If an SCO link is established and not all slots are occupied, the master can at the same time establish an ACL link to any other slave. That is also possible for the slave with the SCO link.

4.8.1 ACL links

The ACL link provides a packet-switched connection between the master and all of the piconet participating slaves. At most one ACL link can exist between a master and a slave. Packet retransmission is applied to assure data integrity.

A slave is always allowed to return an ACL packet in the following slave-to-master slot.

If no data needs to be sent on the ACL link, no transmission shall take place.

4.8.2 SCO links

On an SCO link the master has the possibility to reserve time slots. The link can be considered as a circuit-switched connection between the master and the slave. These links can be used to support time-bounded information like voice.

A single master can support up to three SCO links at the same time. These links can be connected to a single slave or to different slaves. Since there is no time to correct real time voice messages, retransmissions are never used in connection with SCO packets. A slave is always allowed to respond to a SCO packet in the direct following slave-to-master time slot.

When a SCO link is used the master will send SCO packets at regular time intervals.

4.9 Packet structure

4.9.1 General

Transmitted data on a Bluetooth channel is conveyed with the help of packets. The standard packet format is illustrated in Figure 4.5 below; the size (in bits) stands next to the fields. The packet consists of three parts: the access code, the header and the payload.

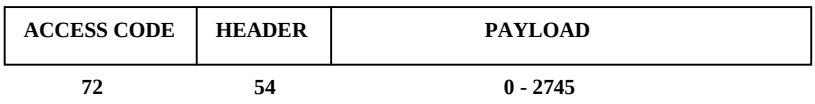

Figure 4.5: Standard packet format with field sizes in bits.

4.9.2 Access code

The 72 bits access code field of a packet is among others used for identification and synchronisation. All packets, which share the same piconet channel, use an identical access code. Almost all packets has an access code, the only exception is the so-called FHS packet. See section 4.9.5 for further information about different packet types.

The access code is also important in connection with inquiry and paging schemes. In this case, the access code has functionality as a signalling message and both the header- and payload field are excluded.

There are three different types of access codes:

- Channel Access Code (CAC)
- Device Access Code (DAC)
- Inquiry Access Code (IAC)

Depending on which operating mode the Bluetooth unit is located in, one of these three different access codes is used.

The *channel access code* identifies a piconet channel and this code is included in all packets, which are exchanged on the channel.

The *device access code* is used for signalling procedures. It could be when a master is paging a slave, or in the other direction, when a slave is responding to master page message.

The *inquiry access code* is used for discovery of Bluetooth units, which are in range.

4.9.3 Packet header

The packet header consists of six different fields and their total size are 18 bits and after protection with a FEC the total size become 54 bits, see Figure 4.5 above. The fields contain information, which are used for controlling the link. Figure 4.6 and Table 4.1 below give further information about the header.

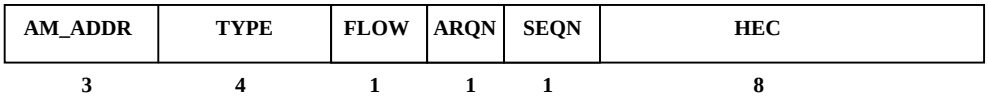

Figure 4.6: Format of packet header.

Field name	*Size (in bits)*	*Explanation*
AM_ADDR	3	Active member address
TYPE	4	Packet type code
FLOW	1	Flow control
ARQN	1	Acknowledge indication
SEQN	1	Sequence number
HEC	8	Header Error Check

Table 4.1: The six different fields in the packet header.

Every active slave has an own temporary 3-bit address, called **AM_ADDR**. This address is used to identify each slave. All packets exchanged between the master and the slave, in both directions, always carry the AM_ADDR of the actual slave. The all-zero address is used for broadcasting from the master to the slaves.
A disconnected or parked slave loses its AM_ADDR and a new address has to be assigned when the slave enters the piconet again.

There are totally sixteen different types of packets, which can be sent on a link. The 4-bit **TYPE** field specifies which type the packet has and its code belongs either to an SCO packet or an ACL packet. The TYPE code also reveals how many slots the current packet will occupy.

The 1-bit **FLOW** field is used for control of the packet flow. If this bit is set to 0 it is equal to a STOP indication. Then the only packets that can still be received are control packets, i.e. ID, POLL and NULL packets. SCO packets can also be received even as the STOP signal is set. When the receiver buffer is empty again the ACL traffic can continue and that is indicated with FLOW bit set to 1, a GO indication.

The 1-bit **ARQN** field is used to inform the sending source about the packet transmission. A positive acknowledgement (ARQN = 1) indicates that the reception of the packet payload was successful, otherwise a negative acknowledgement (ARQN = 0) is returned.

The 1-bit **SEQN** field provides a sequential numbering scheme. The scheme takes care of possibly duplicated packets, which can arise when retransmissions are used.

The 8-bit **HEC** field is a header-error-check word, which protects the packet header.

4.9.4 Payload field

The payload field of a packet can contain different types of information. The size can vary from 0 to 2745 bits. If a packet does not occupy all of the possible 2745 bits the transmission time for the actual packet will be decreased.

4.9.5 Packet types

4.9.5.1 General

There are two physical links defined for Bluetooth: the ACL link and the SCO link. For each of these links it is possible to define 12 different packet types. Four

of these packet types are control packets, and they are common for both of the link types.

A 4-bit TYPE code, which can be found in the packet header, specifies the packet type on the different links. The packets can occupy one, three or five time slots. Table 4.2 below shows the defined packet types. Some of the fields are undefined and can be used for future implementations.

TYPE code	*Slot occupancy*	*ACL link*	*SCO link*
0000	1	NULL	NULL
0001	1	POLL	POLL
0010	1	FHS	FHS
0011	1	DM1	DM1
0100	1	DH1	Undefined
0101	1	Undefined	HV1
0110	1	Undefined	HV2
0111	1	Undefined	HV3
1000	1	Undefined	DV
1001	1	AUX1	Undefined
1010	3	DM3	Undefined
1011	3	DH3	Undefined
1100	3	Undefined	Undefined
1101	3	Undefined	Undefined
1110	5	DM5	Undefined
1111	5	DM5	Undefined

Table 4.2: Different packet types defined for ACL and SCO links.

4.9.5.2 Common packet types

Under this section common packet types for ACL and SCO links will be described. The following five packet types are common: *ID packet*, *NULL packet*, *POLL packet*, *FHS packet* and *DM1 packet*.

4.9.5.2.1 ID packet

The ID packet, or the identity packet, has a fixed length of 68 bits. It can just carry a device access code (DAC) or an inquiry access code (IAC), i.e. no packet header or payload is represented in this packet. The ID packet is for example used in paging, inquiry, and response routines.

4.9.5.2.2 NULL packet

The NULL packet has a fixed length of 126 bits. It contains only the channel access code (CAC) and the packet header, i.e. no payload in this packet. The NULL packet is used for carry acknowledgement when link information is sent. It can for example be information that tells about the previous transmission's success (ACK) or failure (NAK). Flow control information can also be carried.

The NULL packet itself does not need to be acknowledged.

4.9.5.2.3 POLL packet

The POLL packet is similar to the NULL packet, but in contrast the slave must always acknowledge it with some packet. The packet can be used when the master will "poll" the surrounding slaves in the piconet. Then the slaves have to respond, even if they do not have anything special to send.

4.9.5.2.4 FHS packet

The FHS packet is a special control packet and is for instance used for revealing the Bluetooth device address and the clock of the sender. To protect the 144 bits of payload information, a 16 bits CRC code is used. The payload is also coded with a rate 2/3 FEC (see section 4.10.2), which gives payload a total size of 240 bits.

The FHS packet is used for frequency hop synchronization before a piconet has been established, or when an existing piconet changes to a new piconet.

4.9.5.2.5 DM1 packet

The DM1 packet is used for sending control information in any link type, but can also be used for carrying regular user data. The DM1 packet can interrupt synchronous information on an SCO to send control information, as control information has higher prior than the information sent on the SCO link.

4.9.5.3 ACL packets

ACL packets are carried on asynchronous links and contain either ordinary user data or control information. All of the ACL packets contain a CRC code and retransmission is used to secure the transport of the information. The only exception is the AUX1 packet, which does not have any CRC code. Retransmission is never applied for AUX1 packets.

Some of the ACL packets also use FEC codes.

In Table 4.3 below, further information about the different ACL packets is represented.

Packet name	*Payload header size (bytes)*	*User payload size (bytes)*	*Slot occupation*	*CRC code*	*FEC code*
DM1	1	0-17	1	Yes	2/3[2]
DH1	1	0-27	1	Yes	No
DM3	2	0-121	Up to 3	Yes	2/3[2]
DH3	2	0-183	Up to 3	Yes	No
DM5	2	0-224	Up to 5	Yes	2/3[2]
DH5	2	0-339	Up to 5	Yes	No
AUX1	1	0-29	1	No	No

Table 4.3: ACL packets.

4.9.5.4 SCO packets

SCO packets are used on synchronous links. Retransmissions are never applied to SCO packets while the carried data contains speech information. One exception is the DV packet that can contain both data and voice information. Some of the SCO packets also use FEC codes.

The SCO packets are defined to handle 64 kb/s speech transmissions.

In Table 4.4 below is more information about the different SCO packets represented.

Packet name	*Payload header size (bytes)*	*User payload size (bytes)*	*Slot occupation*	*CRC code*	*FEC code*
HV1	-	10	1	No	1/3[2]
HV2	-	20	1	No	2/3[2]
HV3	-	30	1	No	No
DV	1[3]	10 + (0-9)[3]	1	Yes[3]	2/3[3]

Table 4.4: SCO packets.

[2] See section 4.10 below for further information about the meaning of the FEC codes.
[3] The actual information belongs to the part of the packet that contains the data information.

4.10 Error correction

Bluetooth supplies the three following error correction schemes:

- 1/3 rate FEC
- 2/3 rate FEC
- ARQ scheme used for data

Use of an FEC scheme to protect the data payload will cause some overhead, but at the same time this technique will reduce the total number of necessary retransmissions. If the environment is rather error-free and FEC is used, the throughput of interesting data will be reduced. Different packet types with different FEC rates can therefore be useful to have available.

The packet header is always protected by a 1/3 rate FEC to guarantee better transmission of valuable link information.

Section 4.10.1 and 4.10.2 below describe the FEC schemes and section 4.10.3 describes the ARQ scheme.

4.10.1 FEC scheme with 1/3 rate

When the 1/3 rate FEC scheme is used every bit is repeated three times. The packet header and the HV1 packet use this scheme. See Figure 4.7 below.

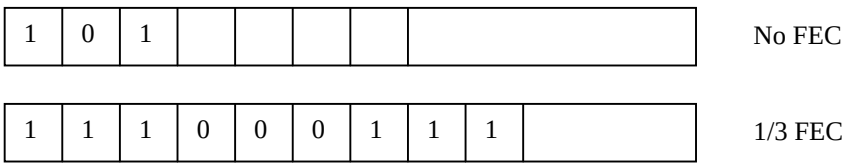

Figure 4.7: Bit sequence with and without FEC.

4.10.2 FEC scheme with 2/3 rate

The 2/3 FEC scheme uses a so-called (15,10) *Hamming code* to generate a new encoded word. 10 bits a time is fed into a generator and the result is a 15-bit coded word and thus the name 2/3 rate FEC. The length of the data that shall be encoded must always be a multiple of 10.

4.10.3 ARQ scheme

DM, DH and DV packets use the ARQ scheme to guarantee that packets are delivered successfully. Each packet is transmitted until a positive acknowledgement is received or a timeout is exceeded. This acknowledgement can be delivered in the packet header of the next returning packet.

The ARQ scheme only works for the data payload of the packet and does not include the packet header.

4.11 Defined states

4.11.1 Standby state

A Bluetooth unit has the *standby state* as default state. The power consumption is extremely low in this state, as only its own clock is running. During fixed time intervals the unit must wake up and leave the standby mode to be able to scan for inquiry or page messages. If the unit answer, with an incoming page message it will not return to the standby state, instead it will enter the connection state as a slave. The unit, which transmits the page message, will get connected as the master.

4.11.2 Connection state

When the connection state has been reached, traffic can flow between the master and the slave(s). It is the master access code and the master Bluetooth clock, which decides the unique channel-hopping scheme.

Bluetooth units can work in several different modes during the connection. The modes are: *active mode*, *sniff mode*, *hold mode* and *park mode*. Theses modes are described in the following sections.

4.11.3 Active mode

When a Bluetooth unit is in *active mode* it uses the channel rather regularly. The slave is assigned a unique active member address when entering the active mode. It is always the master that schedules the transmission to the slaves involved, based on the current traffic condition.

Active slaves always listen in the master-to-slave slot to search after packets. If a packet contains the address of the actual slave it must be ready to receive the content of the packet, otherwise the slave can go back to sleep until the next

master-to-slave slot. The number of occupied slots can be recognised from the TYPE field of the packet.

A periodic master transmission is required to keep the slaves synchronised to the channel. A slave can synchronise itself to the channel with help using the channel access code, which can be recognised in all transmitted packets.

4.11.4 Sniff mode

A slave can be put into *sniff mode* when it is in the connection state. This will reduce the activity by the slave compared with the active. In the active mode the slave has to listen to all master-to-slave slots, but in sniff mode the listen interval is reduced to be every D_{sniff} slot. The time slots where the master can start transmission to the actual slave will also be reduced. These so-called sniff slots are spaced with regular time intervals called T_{sniff}.
The sniff mode can only be entered if the slave participates on an ACL link.

4.11.5 Hold mode

During the connection state, an ACL link can be put into *hold mode*. This means that the slave temporary does not support ACL packets on the actual channel. SCO packets will still be supported if there is any ongoing SCO traffic. The slave keeps its active member address.

Hold mode makes it possible to release capacity and for example the slave can connect to another piconet.

The slave can return to the active mode after a certain time interval, which the master and the slave have agreed on.

The hold mode is a low-power mode.

4.11.6 Park mode

If the slave does not need to participate on the piconet channel, but still wants to remain synchronised to the channel, the slave can enter *park mode*. When entering park mode the AM_ADDR is realest. Instead, it receives two new addresses: Parked Member Address (PM_ADDR) and Access Request Address (AR_ADDR).

The PM_ADDR is used to distinguish different parked slaves from each other. The master uses this address when the slave shall become active again.

The slave uses the AR_ADDR if it will try to get active again.

Parked slaves wake up regularly to get synchronised to the piconet channel.

The park mode is a low-power mode and it is used to let more than seven slaves get connected to the same piconet. Notice that only seven slaves can be active at the same time.

4.12 Power management

4.12.1 Features which reduce the power consumption

The Bluetooth system is intended to be adaptable to many different kinds of devices, especially small hand-held devices with their own power source. For this reason it is important that the power consumption can be adjusted depending on the required resources.

There are features included in the Bluetooth system to ensure low-power operations. See the following subsections.

4.12.1.1 Packet usage

The power consumption can be minimised through usage of different packet types, depending on what needs to be sent. For example, if only control information needs to be exchanged on the piconet channel, NULL packets will be used. The NULL packet is a very short packet and reduces the transmission time, which results in reduced power consumption.

Another example is when a unit receives a packet access code. If the packet header HEC fails, the unit will immediately return back to sleep again and ignore the rest of the packet. This feature also reduces the power consumption.

4.12.1.2 Slot occupancy

When a unit receives a packet the intended receiver can be determined with help of the active member address. If the packet is not intended for the actual slave it goes back to sleep again, for the remaining slots the packet may occupy.

4.12.1.3 Low-power modes

There are three modes, earlier described under section 4.11, which also reduce the power consumption.

4.12.2 Current consumption

There are two major modes in which Bluetooth can operate: *standby* and *call mode*. The standby mode can be divided into four different minor modes: *sleep*, *page-scan*, *hold* and *park mode*. The call mode can also be divided into different minor modes, these are: RX, FH and TX. These modes will influence the current consumption while the link is up.

In *sleep mode* the entire Bluetooth link is deactivated and it is only the low-power frequency oscillator, which is running. Every 1.28 s the link enters *page-scan mode* to search for incoming calls. The scan period is 9 frames long, which correspond to 11.25 s. If an incoming call is detected, Bluetooth enters *call mode*, otherwise it goes back to sleep mode.

In *park* and *hold mode* the link wakes up in regular intervals to scan for frames so the Bluetooth unit can be synchronised to the master.

In *call mode* the baseband and the microprocessor circuits always have to be activated. On the other hand, the radio circuit only works when there is something to transmit. That will affect the current consumption in a good way.

The power consumption in the standby mode can be estimated to be the sum of the sleep and the receiving current, i.e.

$$I_{STANDBY} = I_{SLEEP} + I_{RX} = 0.03 + 0.3 \approx 0.3\ mA$$

The power consumption in the call mode can be estimated to be the sum of the receiving current, the frequency-hopping (FH) synthesiser and transmitting current, i.e.

$$I_{CALL} = I_{RX} + I_{FH} + I_{TX} = 10 + 10 + 10 \approx 30\ mA$$

See also Table 6.6 for further information about power consumption in different modes.

4.12.3 Power consumption by involved processor

To control a Bluetooth unit a processor must be present. This processor can either be integrated with the Bluetooth chip or work like an external processor. In both cases the power consumption by the processor will be affected by the activity of the Bluetooth unit. If the Bluetooth unit is put into park or hold mode the processor does not need to work so hard. Only when the Bluetooth unit wakes up on its regular intervals, to synchronise to the piconet channel, the current consumption will increase for a short time interval.

In a similar manner the current consumption will also be affected of the traffic situation when a Bluetooth unit is put in active mode. If packets are sent all the time the processor will be more busy and that results in increasing current consumption by the processor.

These power consumption aspects are important to take into consideration when a system shall be configured. The power consumption can be very varying dependent on the characteristics of the traffic and that will be noticeable if the Bluetooth units are thought to be supplied by batteries.

5 Profiles

There are 13 different profiles defined for Bluetooth. Each profile defines a selection of messages and procedures, which a Bluetooth device must be able to support. The Bluetooth Specification defines these messages and procedures. A device must not support all profiles but at least one, the Generic Access Profile. Some of the profiles are also dependent on each other.

Among other things the idea with implementing different profiles into different devices is to reduce the needs for powerful processors and large memories. But the reason is also to get a common behaviour among the devices involved. All Bluetooth devices do not need to be able to speak to each other.

All devices must not support everything described in a profile. Some characteristics are mandatory and some are optional.

See Appendix F for more information.

6 Simulation model

6.1 Introduction

A relatively big part of this Master Thesis was to figure out what the model of a Bluetooth piconet should look like. Some aspects that the model had to cover were:

- to build up in a modular fashion in accordance with Bluetooth stack
- to be able to simulate up to eight Bluetooth devices
- to be easy to change
- power consumption
- disturbances upon the radio link

A well-written documentation, which describes the functionality and how to use the model, was also a requirement.

When you are implementing a model like this it is important not to look so deep down in all details. If you do that it will be very hard to implement everything. Instead you have to look at the basic functions like letting the master or the slave be able to send and receive packets, have possibilities to make retransmissions and so on, to let the model get a natural flow.

With the implemented model different things can be simulated. The following parameters are common for both data and voice traffic:

- Connection delays
- Packet losses
- State times
- Power consumption
- Total number of transmitted bits

Unique simulation parameters for voice traffic are:

- Blocking probability
- Number of success voice connections
- Number of voice connection attempts

Unique simulation parameters for data traffic are:

- Wanted transmission rate
- Real transmission rate

It is also possible to see the utilisation of the channel.

6.2 Included features from the Bluetooth Specification

Most of the functionality of the implemented model is based on the Baseband Specification of the Bluetooth Specification Version 1.0 A. The Baseband is a low-level layer of the Bluetooth stack and is suitable to use for an implementation. The Baseband Specification describes how the basic flow of a Bluetooth system is thought to work.

The following parts of the Baseband Specification have been taken into consideration and are implemented in the model:

- Physical channel
  - Time slots
  - Bit rates

- Physical links
  - ACL link
  - SCO link

- Packets
  - ID
  - NULL
  - POLL
  - FHS
  - DM1
  - DH1
  - HV3

- Transmit and receive routines
  - ACL traffic
  - SCO traffic

- Retransmissions
  - ARQ scheme

- Channel control
  - States
    - Active
    - Park
    - Sniff

  - Procedures
    - Inquiry[4]
    - Page[3]

Parts from the Link Manager Protocol have also been used to get realistic transmission behaviour.

6.3 General description of what a user can do with the implemented model

The implemented model can handle simulation of a piconet, i.e. one master and a number of slaves. The number of slaves, which can be present at the same time, is restricted to 19. In reality, the number of slaves can be many more, but changes that need to be done to allow additional slaves to be present are not so complicated. See Appendix C for more information about that.

Two different types of traffic models can be simulated, these are:

- slaves which want to set up a duplex voice connection.

- slaves which want to set up a file transfer.

The user can set different characteristics of a slave. Some of the choices, which can be made before the simulation starts are:

For data traffic:

- transmission file size
- request interval
- transmission rate
- packet type[5]
- etc

[4] Inquiry and page procedures are implemented for an eventual expansion of the model, but are not in use.
[5] DM1 and DH1 packets are supported in the model.

For voice traffic:

- call length
- request interval
- packet type[6]
- etc

See also Figure 6.1 below, which is an example of the implementation in OPNET.

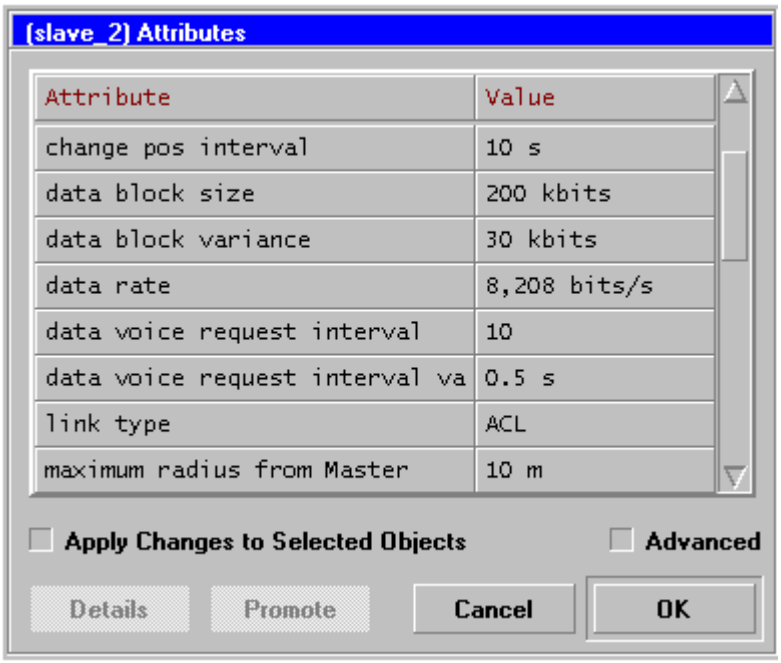

Figure 6.1: Slave attributes.

The user can also, place a disturber among the other devices. The disturber will affect the transmissions negatively, as more packets will get lost. The user can decide how the disturber shall act, i.e. in form of movement, transmitting effect, etc. See section 9.2.3.

After a simulation several different results can be studied in the form of different graphs. Some of the possible results are given below:

- State times

[6] HV3 is the only implemented packet type for voice traffic. Further packet types can be added.

- Energy consumption for different states
- Expected and real transmission rate
- Lost packets
- Delays
- Utilisation of the channel
- Blocking probability
- The disturber's movement
- etc

See also Figure 6.2 below. It is a screen shot from the implementation in OPNET.

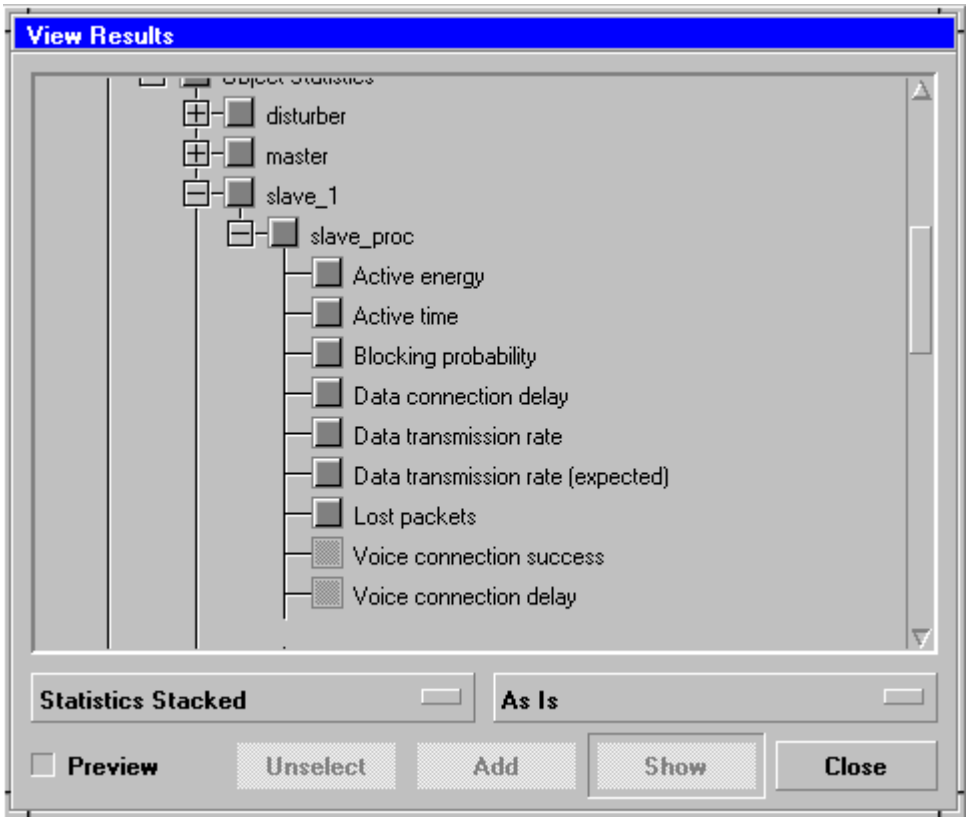

Figure 6.2: Simulation results.

6.4 The extent of the model

In the following subheadings the extent of the model and the implementation will be discussed.

6.4.1 Description of the implementation

The implementation is done with the following three different process models:

- the master's process model
- the slave's process model
- the disturber's process model

Most of the functionality is located at the master, who controls the traffic. The slaves are able to influence the ongoing traffic but it is always the master who decides if a slave can be active on the channel. The only way the disturber affects the simulation is by changing its position with a regular time interval, which will affect the received signal strength by either the master or the slave.

6.4.1.1 The master's process model

Under this subtitle following parts are included:

- Answer of requests from slaves
- Link support
- Packet losses
- Retransmissions
- Park and unpark procedures
- Built-in timer function
- Sniff mode traffic

The master is always waiting for slave requests and cannot generate its own requests. If there is enough capacity to let another slave join the ongoing traffic on the channel, the master starts to *unpark* the slave. The unpark procedure consists of a few packet transmissions between the master and the actual slave. After this procedure is finished the slave is put into active mode and can soon receive data or voice packets.

If the new link connection is an ACL link the master starts to transmit the actual ACL packet type and waits for acknowledgements. The implementation supports DM1 and DH1 packets, which are useful for data transfers. See section 4.9.5.3 for more information about these packets.

If the new link connection is an SCO link the master starts to transmit SCO packets to the slave. The voice connection assumes to be duplex and therefore the slave also returns SCO packets. The implementation supports HV3 packets.

Every time the master receives a packet, the implemented functions calculate if the actual packet has been lost during its transmission between the slave and the master. If the result of the transmission is a packet loss, this is noticed and statistics is being written to a special OPNET file.

When the data or voice transmission is finished, a *park* procedure starts to park the slave. During both the unpark and park procedure other slaves maybe have to wait until the unpark or park packet sequence is finished. The link control information assumes always to have higher prior than ordinary traffic.

The master has a timer function implemented. This means that the master tries to do retransmission if the transmitted packet consists of link information. If it is not link information that is being sent, a counter controls the retransmission function.

The master can also handle sniff mode traffic. Some of the involved slaves maybe not need to participate all the time. This means that the master only schedules transmissions to the slave at some fix time intervals. This time interval can be affected by the slave's attribute called *sniff interval*.

The master has several attributes, which can be set before the simulation starts. To get more information about these attributes, see section 9.2.1.

6.4.1.2 The slave's process model

Under this subtitle the functionality of a slave will be discussed. The following parts are included:

- Connection requests and pattern
- Link support
- Packet types and sizes
- Built-in timer function
- Retransmissions

When a simulation starts all slaves begin in park mode. Depending on which characteristics the different slaves have been given, they will at randomly chosen times ask the master if they can join the network.

If it is an ACL connection for ordinary file transfer, a function will calculate how many packets are needed for the actual file size. The number of packets is also dependent on packet type that is being used. The packet type can be chosen before the simulation starts with the slave's attribute *packet type*. Different packet types can carry different payload sizes. The mean value of the file size can be set with one of the slave's attributes called *data block size*.

If the master is busy the slave tries to do new connection requests until it success.

If it is an SCO connection, the transmissions between the master and the slave will go on until a specified time. The mean value of the call length can be set with one of the slave's attributes called *voice connection time*.

The slave has like the master a built-in timer function for retransmissions of link information packets. If an ordinary packet of a file transfer is dropped during its transmission from the master to the slave, no timer function will expire. Instead, a counter, which controls the number of successful received packets, takes care about eventual losses. This means that the transmission will continue until the whole file is received. SCO connections do note use retransmission at all.

If the slave cannot get a voice connection immediately it will try to do a new request after the actual timer has expired. The time between the next coming request is made randomly, and the time interval is varying between 0 and 3 seconds. The slave tries to do a new requests up to three times, then the slave waits a longer period, according to its request interval, before the next request is done.

The slave has many more attributes, which can be set before the simulation starts. To get more information about these, see section 9.2.2.

6.4.1.3 The disturber's process model

Under this subtitle the functionality of a slave will be discussed. The following parts are included:

- Movement of the disturber
- Antenna gain and effects

The disturber is a rather passive unit, which can affect the transmissions on the channel. The slaves and the master will receive different signal strengths depending on the position of the disturber. Its position is changed randomly with a function, which uses a normal distribution to calculate the disturber's next position. These movements can be affected by the disturber's attribute called *movement*. The time interval between position updates can be controlled by the slave's attribute *change pos interval*.

The attributes *antenna gain* and *transmitting effect* can be changed and these will affect the signal-to-interference calculations, which is done during the simulation.

The disturber is given a start position with the help of the following two attributes: *jammer x position* and *jammer y position*. The movement of the disturber is restricted to a maximum radius, which cannot be passed. The radius is measured between the disturber and the master, which position is fixed during the whole simulation. The disturber's attribute *max movement radius* set up this restriction.

The *status* attribute turns on and off the disturber function.

6.4.2 Representation of Bluetooth time division channel

The time unit that represent a time slot in the Bluetooth system is 0.000625 seconds long. This is the shortest unit that the model uses for different calculations. To get a correct behaviour it is important that the transmission time is exact. To guarantee that a reference parameter is set when the simulation starts. If necessary, this parameter can be used to decide the actual time slot.

The different frequency channels that the Bluetooth system uses are not represented in the model. It is not so interesting when only one piconet is under investigation.

The frequency dependence would be of more interest if several piconets were present, since then interference from other piconets could be possible to evaluate.

6.4.3 Movement of modules

In the model, the master of the actual piconet is assumed to have a fixed position. This position can only be set before the simulation starts. The position is determined by an x-coordinate and a y-coordinate.

Both slaves and the disturber can in contrast to the master change their positions during the simulation. See section 6.4.3.1 and 6.4.3.2 below.

6.4.3.1 Movement of the Disturber

The movement of the disturber during a simulation is based on a random function. The disturber is assigned a maximum working radius before the simulation starts. The random function generates one difference for the x-coordinate and one difference for the y-coordinate. After addition of these differences to the previous position a new position is generated.

Every time an update of the disturber's position is made, a check of the new distance between the master and the disturber is done. If the new radius is larger than the accepted one, a new position is generated. This procedure is repeated until an allowed position is found. The new position is always based on the previous position, i.e. the position the module had before the position update started.

The disturber position update interval can be set before the simulation starts.

6.4.3.2 Movement of the a slave

The movement of a slave is similar to the disturber's, i.e. it is based on a random function. The movement difference is thought to be smaller than the disturber's movement difference, since the coverage area for the slaves is restricted. By default the maximum movement radius is set to 10 meters. If the user increases this distance to be greater than 10 meters it will only affect the signal-to-interference ratio when packet loss calculations are done. See section 6.4.4 for more information about packet loss calculations.

The slave position update interval can be set before the simulation starts.

6.4.4 Interference upon the radio link

Every time a packet is received either by the master or by a slave, there is a possibility that the packet has been affected during its transmission. The model considers two different types of interference, which can occur. The first is a *basic interference* that depends on the distance between the master and for the moment transmitting slave, see section 6.4.4.1 below. The second type of interference that can affect the packet transmission, depends on an eventual surrounding disturber, see section 6.4.4.2 below.

6.4.4.1 Interference depending on distance between master and slave

During the implementation some measurements have been done with Ericsson's Bluetooth Development Kit (see Appendix H), to give a rather rough estimate of how the distance between the master and the slave affects the packet transmission. The reason why some packets are lost even when no active disturbers are in the neighbourhood depends on the environment. Multi-path propagation is a dominant factor and can sometimes cause failures. Let us call these failures *basic disturbances* and observe that they can behave differently depending on location.

An application that is included in Ericsson's Bluetooth Development Kit makes it possible to send AUX1 packets between the master and the slave. The AUX1 packet does not include either CRC code or FEC code. The transmitted data payload cannot therefore be corrected in any way. See also section 4.9.5.3 for information about AUX1 packets.

The information that can be interesting after such a transfer is the BER (Bit Error Rate) and number of received packets. In the model, just the number of lost packets has been chosen to calculate the packet loss probability. The implementation makes no consideration about bit errors.

The following three different distances have been used to measure the number of lost packets: 1.5, 4.0 and 5.5 meters. The reason why exactly these distances has been chosen depends only on the restrictions of the area of room, where the measurements where done. The maximum coverage radius with an 0 dBm antenna is as earlier mentioned about 10 meters.

For every distance 10,000 packets has been sent between the master and the slave a couple of times. These transfers have then been repeated ten times to be able to get a correct mean value of the number of lost packets. That is, totally 100,000 packets per distance has been transmitted. Notice that the output effect from a Development Kit has a fixed value of 1 mW (effect class 3). Later when the Bluetooth technology is implemented in real products, the output effect will be adaptive. That is, communicating devices will try to reduce the output effect as much as possible.

The results of the measurements, for the three different distances, are shown in Table 6.1, Table 6.2 and Table 6.3 below.

Transmission	***Lost packets***	***BER (ppm)***
1	7	1277.62
2	27 (7)	1648.85
3	6	1190.37
4	0	936.21
5	0	996.55
6	1	962.60
7	0	997.41
8	1	1004.41
9	3	950.72
10	0	951.72

Table 6.1: Transmission of AUX1 packets between master and slave. Distance 5.5 meters.

Due to table 6.1, a packet loss of 27 packets is shown for the second transmission. This value is extremely high compared with the other values and maybe it should be ignored. The relative high rate of packet losses compared to the other transmissions at this distance could have been a result of a temporary disturbance. However, in the implementation this value is included and that might have affected the results. But the calculated packet loss probability (see Table 6.4 below) for the different distances is very small and therefore I do not think it could have affected the simulation results so much. Instead, the packet loss caused by the disturber has a more distinct influence on the results.

I have chosen to calculate with the measured value, because it is very difficult to prove what's affecting the second transmission. If any changes shall be done the actual value can be estimated to 7. That assumption is based on the other

measurements in Table 6.1. This will result in a packet loss probability of 0.00025.

Transmission	*Lost packets*	*BER (ppm)*
1	1	1092.78
2	2	1075.65
3	0	1113.36
4	0	909.48
5	2	920.44
6	3	1017.98
7	0	918.97
8	0	964.22
9	1	961.73
10	0	967.24

Table 6.2: Transmission of AUX1 packets between master and slave. Distance 4.0 meters.

Transmission	*Lost packets*	*BER (ppm)*
1	0	517.24
2	0	528.02
3	1	510.83
4	0	513.36
5	1	511.25
6	0	507.33
7	1	506.52
8	0	512.07
9	0	518.53
10	0	535.34

Table 6.3: Transmission of AUX1 packets between master and slave. Distance 1.5 meters.

The calculations to determine the loss probability for the respective distances are rather simple. The total number of lost packets divided with the total number of transmitted packets, in this case 100,000 packets, gives the result. See equation 6.1 below.

$$\textit{Packet loss probability} = \frac{\textit{Total number of lost packets}}{\textit{Total number of transmitted packets}} \qquad (6.1)$$

The results are illustrated in Table 6.4 below.

Distance between master and slave	*Packet loss probability*
5.5	0.00045 (0.00025)
4.0	0.00009
1.5	0.00003

Table 6.4: Packet loss probabilities depending on different distances.

6.4.4.2 Interference depending on a surrounding disturber

The model includes a one-band disturber, which can be representative during a simulation. The user has the possibility to activate or inactivate the disturber before the simulation starts.

The disturber assumes use of a fixed frequency that during some time slots can interfere with the ongoing traffic of the actual piconet. This means that the frequency, which is located in the ISM band, has the possibility to collide with one of the previously described channels. See section 3.2 for further information about these channels.

The probability calculations the implemented disturber uses are done with the assumption that 79 different hop channels are used and that will affect the collision probability. At the same time, the signal-to-interference is the dominant factor that decides if a packet loss shall occur.

Of course, the probability will depend on the actual utilised capacity of the piconet. For example, if only 50 per cent of the time slots are used the probability for interference will be reduced to the half.

The model assumes that all 79 different channels are equally utilised, and that seems to be correct if you study the traffic for a longer time interval. Then it is easy to realise that a fix number of time slots will be able to interfere during the assumed interval, and that gives a probability of *0.013*. See the calculations below.

Probability for time slot interference:

$$\frac{1}{\text{Frequency hop rate}} = 1 / 1600 \approx 0.013$$

Then, if the actual traffic load is reduced the probability for interference will decrease. This can be described as follows:

Probability for time slot interference with reduced traffic load:

*Traffic load * 0.013*

The parameter *traffic load* can have a varying value between 0 and 1.

If the distance between the receiving slave (or master) and the surrounding disturber is large enough, the signal from the disturber will be too weak to cause any trouble. Therefore the received signal strength from both the sending slave (or master) and from the disturber is measured for all transmitted packets.

In compliance to the Bluetooth Specification [1] the actual limit for the S/I (signal to interference) is derived to be 11 dB. If the S/I level is below that limit the carried information will not be useable.

The formula used in the model for calculation of S/I looks as follows.

$$S / I = P_{RX\ MASTER} / P_{RX\ DISTURBER} \qquad (6.2)$$

In this case the signal-to-interference ratio is calculated for signals received by the slave. The formula can also be used for calculations of signal-to-interference ratio by the master if $P_{RX\ MASTER}$ is changed to $P_{RX\ SLAVE}$ instead.

The two needed effects $P_{RX\ MASTER}$ and $P_{RX\ DISTURBER}$ are depending on four other parameters: P_{TX}, G_{TX}, G_{RX} and L_{FS}. Their relationships, based on this example, are shown in equation 6.3 and 6.4 below. All values are represented in dB.

$$P_{RX\ MASTER} = P_{TX\ MASTER} + G_{MASTER} + G_{SLAVE} + L_{FS\ M->S} \qquad (6.3)$$

$$P_{RX\ DISTURBER} = P_{TX\ DISTURBER} + G_{DISTURBER} + G_{SLAVE} + L_{FS\ D->S} \qquad (6.4)$$

A description of the just mentioned parameters is given in Table 6.5.

Parameter	Explanation
$P_{TX\ MASTER}$	Master transmitted effect.
$P_{TX\ DISTURBER}$	Disturber transmitted effect.
G_{MASTER}	Transmitting master antenna gain.
$G_{DISTURBER}$	Transmitting disturber antenna gain.
G_{SLAVE}	Receiving slave antenna gain.
$L_{FS\ M->S}$	Free space signal loss between master and slave.
$L_{FS\ D->S}$	Free space signal loss between disturber and slave.

Table 6.5: Description of parameters.

The outgoing transmitted effect from the master is assumed to be 0 dBm (1 mW), which is correct if power class 3, is used. The outgoing transmitted effect from the disturber can be chosen before the simulation starts and can therefore be assigned different values.

The master antenna gain, the slave antenna gain and the disturber antenna gain can also be chosen before the simulation starts.

The signal loss dependence, L_{FS}, can be calculated with help of equation 6.5 below. The usage of this equation is correct if the transmitting and receiving antennas can be assumed to be *isotropic* antennas (see Appendix E). In the Bluetooth Specification under the first chapter, the Radio Specification, it is assumed that Bluetooth uses isotropic antennas is made.

Free space transmission loss is defined by equation 6.5 below.

$$L_{FS} = 32.4 + 20 * \log f_{MHz} + 20 * \log d_{km} \qquad (6.5)$$

The returned value from equation 6.5 is represented in dB. The unit of the frequency component is given in MHz and the unit of the distance component is given in km. In the model the frequency is fix and a value of 2400 MHz is used, which is a good approximation for the actual frequencies that Bluetooth works with.

Free space transmission loss depends in other words on the actual frequency that is used and the actual distance between transmitting and receiving modules.

6.4.5 Power consumption

Power consumption is a very important part when you are dealing with handheld devices like portable computers, mobile phones, etc. It is therefore interesting to be able to calculate how much current a Bluetooth device will use depending on it's field of application.

The model lets the user affect the power consumption of different states. Power consumption values belonging to the respective states can be set before the simulation starts. There is two default effect values for the active state: 90 mW and 27.6 mW. The first value is a maximum power consumption value that can be used if the Bluetooth unit is assumed to transmit all the time when it is active. The second value represents a Bluetooth unit, which is transmitting with a data rate of 9600 bits per second. Notice that if the data rate attribute of a slave is set to a special value, it will not automatically influence the active state power consumption. The user must set this value separately. See also section 9.2.2.

For the other states, i.e. the park state, the sniff state and standby state no exact effect values have been found. Table 6.6 below shows some approximated value that has been found from a seminar [11]. Therefore it is up to the user to assume some probable values before the simulation starts.

Mode	***Current***	***Effect***
Standby	300 µA	810 µW
Sleep	30 µA	81 µW
Hold/Park	60 µA	162 µW
Active	1 or 100 mA	2.7 - 270 mW
Transmitting	3 – 30 mA	8.1 – 81 mW

Table 6.6: Power consumption in different modes.

The values in Table 6.6 assume that a Bluetooth device uses 2.7 V as power supply.

6.4.6 Connection delays

The delay depends on how long time the slave must wait until it has channel access. The slave always has a fix time interval that allows the slave to send requests to the master. Sometimes the slave has to wait longer before its reserved slot will arise and that can affect connection time.

For voice connections the implemented request pattern acts as follows. The actual slave tries to do requests three times, every request within a random time between 0 to 3 seconds. If the slave does not succeed with the connection, it waits one time interval to its next connection.

For data connections the slave tries to get connected until it success. The delay times can therefore be rather long. An implemented random function gives varying time interval between the requests.

6.5 Verification of the model

Today it is not so easy to compare the behaviour of the implemented model with other implementations since the Bluetooth technology is rather new. But to verify the expected characteristics of the implementation have been reached the following things have been done:

- Simulations in debugger mode to check the behaviour of the packet flow between master and slaves, i.e. concerning timing between packet transmissions.

- Analysis of simulation results.

6.5 Profile relations

The implementation is based on the low-level Baseband Protocol and none of the earlier mentioned profiles (see section 5) are implemented. But thanks to the slave's attributes different characteristics can still be given to the slave. An example is given below.

Imagine a cordless office with Bluetooth printers and fax machines, which can communicate with a Bluetooth access point. The access point is used for two reasons:

- to let the fax machines get connected with the company's telephone net via the access point.

- to let passing by persons with portable computers get connected to the access point to be able to print documents on the local printer.

In this scenario different profiles are involved and the devices have also different characteristics. The characteristics can among others differ in data transmission rates, data sizes and number of connections. Even the connection interval will differ depending on whether the requesting unit is a portable computer or a fax machine.

The above-mentioned characteristics can be simulated in the implemented model. That will give different Bluetooth devices different behaviours and a link to the profiles that Bluetooth uses.

6.8 Common procedure not in use

The master and the slave have implemented procedures to handle inquiry and page routines. These routines are for the moment not in use but can be interesting for coming expansion of the functionality. See the *inquiry* state and the *page* state of the master's process model.

7 OPNET

7.1 Introduction

OPNET Modeler (Optimised Network Engineering Tools) is an event based simulation program delivered by MIL3 Inc. The field of application for OPNET is rather big. The idea behind the tool is to let the user be able to do different types of modelling, simulations and performance analysis of network communication and communication protocols. OPNET can either be installed on Windows NT or on a UNIX system. Here are some examples of what the simulation tool can be used for:

- Study of performance of existing networks and protocols.
- Research and development in communication architectures and protocols.
- Network planning
- Resource sizing

Today OPNET is delivered with many already existing protocols. Many of these built-in models are very detailed and include often the desired functions. Some examples of already implemented models of communication protocols are as follows:

- Ethernet
- IP
- UDP
- TCP
- ATM

OPNET offers also a very rich range of models of routers and switches.

7.2 OPNET's editors and layers

When using OPNET, to build up a model, the user works with different types of *editors*. These editors make it easier for the user to implement and overview the model when it tends to grow. The implementation can be done in such a way that it reminds of the hierarchy of a protocol. Compare it with the OSI model.

OPNET consist of three different layers: the network layer, the node layer and the process layer. Below is a short description of the layers given.

Network layer: represents the topology of the actual network which will be simulated. As an example it can be a server and workstations, which are connected to each other with links.

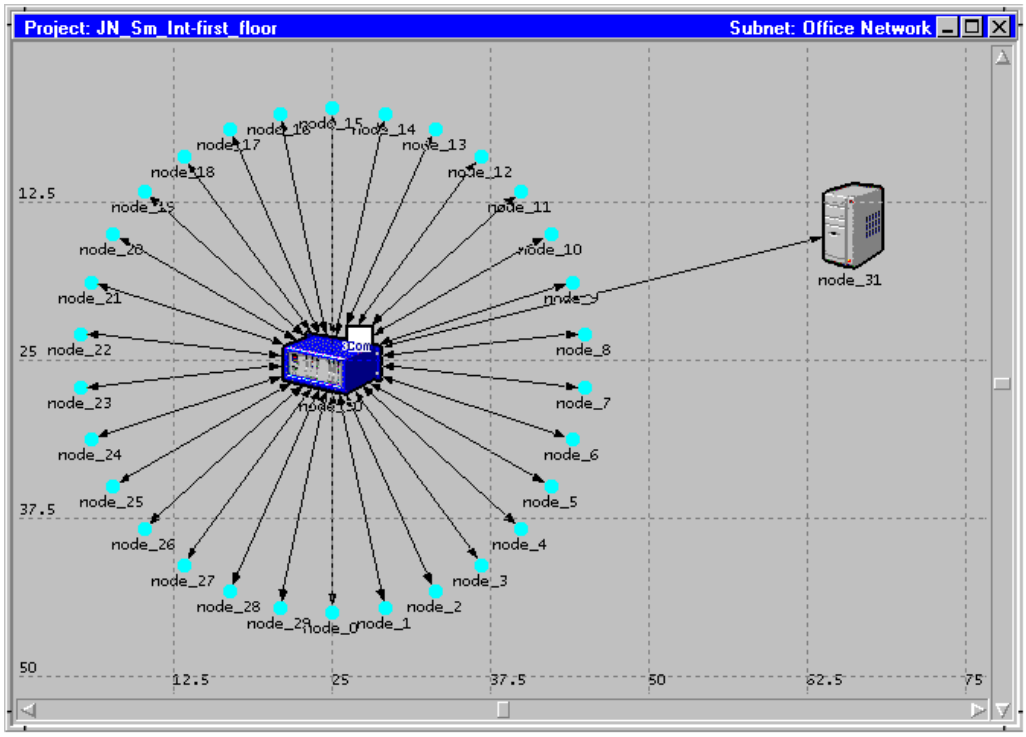

Figure 7.1: Network layer in OPNET.

Node layer: this layer is represented one level below the earlier mentioned network layer. A workstation can for example be represented using this layer. In the Figure 7.2 below, a workstation is shown that can handle IP-traffic. It is built up with several blocks call modules, which represent the protocol stack. Between these modules there are so-called packet links. These can deliver packets to the different layers of the workstation.

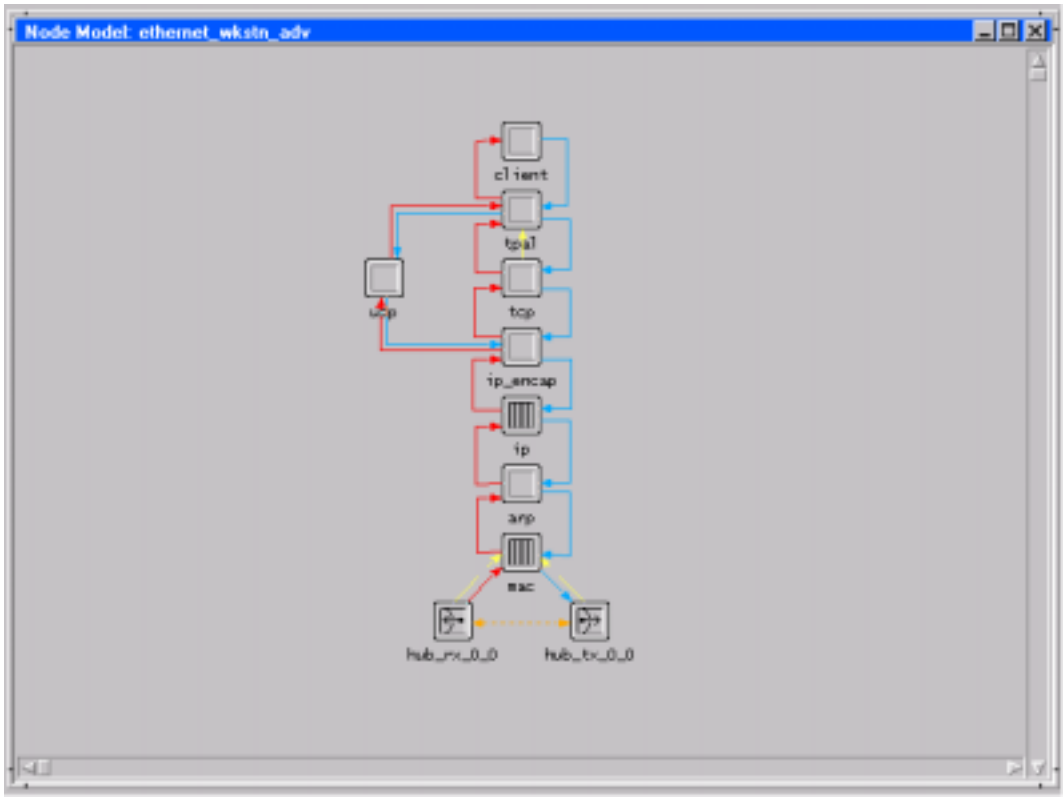

Figure 7.2: Node layer in OPNET.

Process layer: can be found below the earlier mentioned node layer. The purpose of this layer is to define the functionality of the actual protocol. That is done using a state diagram, which can be constructed with this editor. A state diagram consists of one or more defined states and transitions belonging to the state(s). In every state, it is possible to implement characteristics. It can be done with C or C++ code. See Figure 7.3 below.

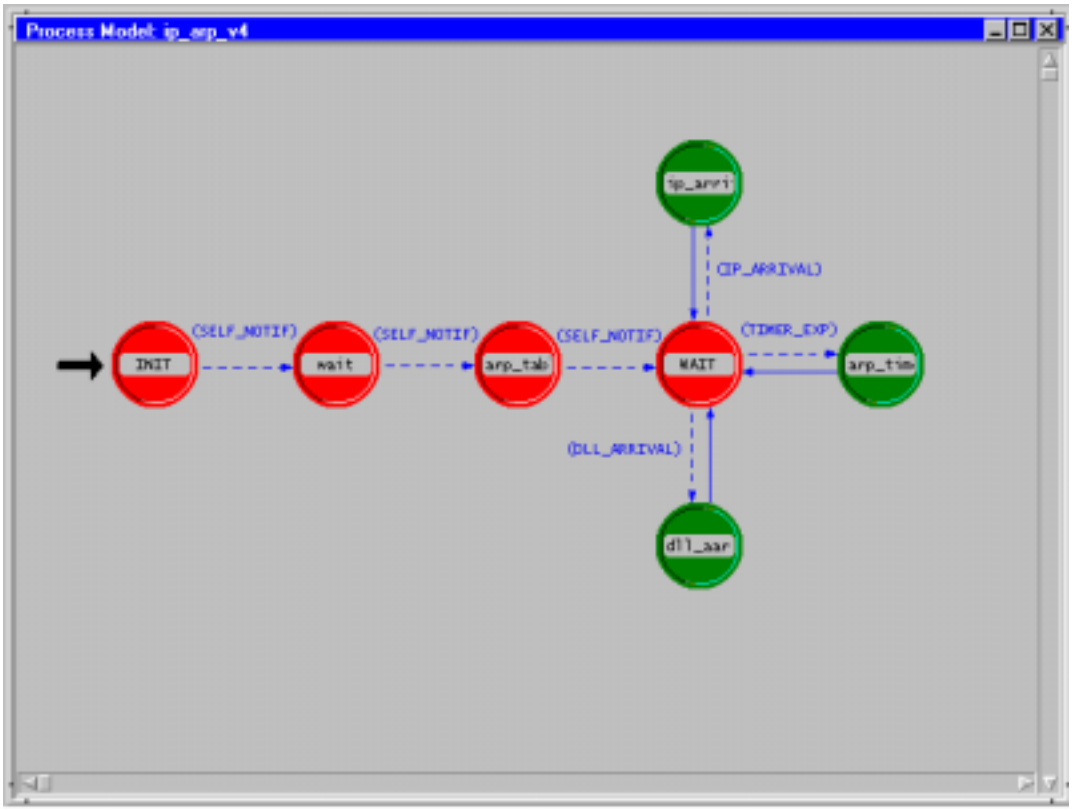

Figure 7.3: Process layer in OPNET.

7.3 OPNET/BONeS – choice of simulation tool

In connection with my implementation of a Bluetooth piconet I had a choice to make: would OPNET or BONeS be the best tool for the implementation. Therefore a part of the work was to analyse which one of these two simulation tools that could be best to use. Now I must also mention that AerotechTelub already had a license for BONeS, but it seemed just to be a question of time before they should order a license for OPNET.

But before that decision was taken I studied the manuals and began to use the different programs. There were guided tutorials for both of the actual tools and that made it rather easy for me to get a feeling for how the different programs worked. Rather soon I realised that OPNET was the tool I wanted to use. See Appendix G for my opinions about the features OPNET offers.

8 Description of the implementation

The implementation is written in C code and the code is located at several places. Most of the code can be found in the different states of the process models, but there are also some other places. A more careful description can be found below.

8.1 Location of the implemented C code

- External include file

Definitions of external parameters and structures can be found in the external file *externvar.c* .

- States of the process model

The defined states contain implemented C code which give the Bluetooth unit its functionality.

- Header block (HB)

The header block contains defined parameters, which are used to identify the different interrupts. The interrupts handle the flow of the model. Function declarations can also be found here.

- Function block (FB)

In the function block the internal functions are defined.

- Static variables (SV)

Here static variables are defined.

- Temporary variables (TV)

Here temporary variables are defined.

8.2 Initiating phase of a simulation

8.2.1 Initiating the master

When the simulation starts the master allocates two different lists which contain *Member_Info* structures. Such a structure itself contains information about a specific slave. Both the master and the actual slave initiate parameters into the actual structure before it is inserted in the *not_active_member_list*. The name and functionality of the different lists are:

- *active_member_list*

To control the slaves which are active.

- *not_active_member_list*

To control the slaves which are inactive.

Other things that happens during the initial phase are:

- The master timer function is initiated.
- Activation of local statistical handlers.

8.2.2 Initiating of a slave

The following things happen during the initial phase by the slave:

- The slave adds some of its simulation attributes to its Member_Info structure.
- Interrupts for position updates and connection requests are scheduled.
- Activation of local statistical handlers.

8.2.3 Initiating of the disturber

The following things happens during the initial phase by the disturber:

- Interrupt for position updates of the disturber is scheduled.
- Activation of local statistical handler.

8.3 Interrupts

Different types of interrupts control the flow of the model. These interrupts can affect a process model itself or process models of other units. For example, a packet reception will give the receiving unit a so-called *stream interrupt*. Depending on the content of the packet the receiving unit can decide what to do.

Every *self* or *remote* interrupt, which belongs to a special slave, can be identified using a code. The code ends with the same number as the actual list position the slave has. That makes it possible for the master to find the slave, which caused the interrupt.

8.4 Statistical updates

Local statistical updates are done when:

- Packets are being sent and received.
- Packets are lost.
- Slaves change their states.
- Slaves and the disturber change their positions.
- Etc

To add further parameters to the implementation see Appendix D.

8.5 Overview of help functions

The following sections describe help functions, which are implemented for the master, the slave and the disturber. A short explanation of what respective function is used for is also given. See Table 8.1 Table 8.2 and Table 8.3 below. If the function is a *print function* it will only be possible to see the results if a simulation is running in *debugger mode*.

8.5.1 Master help functions

Function name	***Description***
Blue_timer_restart (Timer*, double)	Restart of timer.
Blue_timer_start (Timer*, double)	Start of timer.
Blue_timer_stop (Timer*)	Stop of timer.
Change_slave_state (int,int,int)	Change the state of a slave.
Change_state_info (int,int,int)	Print function to get the state of a slave.
Change_traffic_condition (int)	Function to affect the traffic condition. (Not in use)
Check_list_pos (List*)	Print function to view the list content.
Check_sent_pk (Packet*, double)	Print function to view a sent packet.
Dist_between_points (double, double, double, double)	Calculates the distance between two positions.
Find_slave_in_list (List*,int)	Finds the list position depending on the state of the actual slave.
Find_slave_in_list (List*,Objid)	Finds the list position depending on the object id of the actual slave.
Get_free_slot (Member_Info*,int)	Allocates slots in the *slot_vec*.

Get_next_node to_serve ()	Returns the next slave to which a packet shall be sent.
Init_slave_intrpt_vec ()	Initiating of *slave_intrpt_vec*.
Init_slot_vec ()	Initiating of *slot_vec*.
Make_slot_free (Member_Info*)	Set an allocated slot free.
Move_slave_from_active_list (int)	Moves a slave from the *active_slave_list*.
Move_slave_to_active_list (int)	Moves a slave into the *active_slave_list*.
Normal_distribution_outcome (double,double)	Generates a normal distributed value.
Print_slot_index (Member_Info*)	Print function to view allocated slots of the involved slave.
Print_slot_vec ()	Print function to view the content of the *slot_vec*.
Send_fhs_packet (Objid,int)	Function to send FHS packets.
Send_id_packet (Objid,int)	Function to send ID packets.
Send_packet (Member_Info*)	Function to send DM1, DH1, HV3 and NULL packets.
Send_park_packet (Objid,int)	Function to send PARK packets.
Send_poll_packet (Objid,int)	Function to send POLL packets.
Send_sniff_packet (Objid,int)	Function to send SNIFF packets.
Send_unpark_packet (Objid,int)	Function to send UNPARK packets.
Send_unsniff_packet (Objid,int)	Function to send UNSNIFF packets.
Signal_to_inter (double, double, double, double, double, double, double, double)	Function to calculate signal-to-interference.
Update_sent_packet (int,int)	Updates sent bits depending on packet type.

Table 8.1: Master help functions.

8.5.2 Slave help functions

Function name	***Description***
Blue_timer2_restart (Timer*, double)	Restart of timer.
Blue_timer2_start (Timer*, double)	Start of timer.
Blue_timer2_stop (Timer*)	Stop of timer.
Change_loss_prob ()	Changes the packet loss probability.
Check_sent_pk_from_slave (Packet*, double)	Print function to view a sent packet.
Collect_times ()	Function to collect time

	statistics.
Distance_between_ponits (double, double, double, double)	Calculates the distance between two positions.
Find_myself_in_list (List*,Objid)	Finds the list position of a slave.
Generate_data_rate (Member_Info*)	Generates different transmission rates for data connections.
Generate_next_connection (double,double,int)	Generates next connection time and connection length of a voice connection.
Generate_no_of_packets (int,int)	Generates needed packets depending on actual packet type.
Generate_random_number ()	Random number generator.
Get_no_of_active_slave_in_list (List*)	Returns the number of active slaves.
Init_slot_index (Member_Info*)	Initiating of a slave's *slot_index* vector.
Init_times ()	Initiating of state times.
Norm_dist_outcome (double, double)	Generates a normal distributed value.
Print_connections ()	Print function to view the length of a new voice connection.
Print_state_times ()	Print function to view state times of a slave.
Send_slots (double)	Print function to view the number of needed slots.
Set_packet_loss_prob ()	Change the packet loss probability.
Signal_to_interference (double, double, double, double, double, double, double)	Function to calculate signal-to-interference.

Table 8.2: Slave help functions.

8.5.3 Disturber help functions

Function name	***Description***
Normal_dist_outcome (double,double)	Generates a normal distributed value.

Table 8.3: Disturber help functions.

8.6 Placement of devices

The master must always be placed on the project plan first of all units, else an error will occur when the simulation starts. The implementation is dependent on that the master is being initiated first of all units.

8.7 Timers

Both the master and the slave use the built-in timer function. A timer is activated every time the master or the slave is transmitting packets containing state information, i.e. when a slave is going to change its state. The timer function makes it possible to let retransmissions be done when packets are lost.

Retransmissions in connection to ordinary data transfers do not use this timer function, instead counters control the traffic. If a packet is lost during the transmission the master's counter (for the actual slave) will not be increased. That will result in a retransmission of the lost packet.
Notice that retransmissions are never used for SCO packets.

8.8 Resource allocation

All scheduling of traffic is done by the master, which uses a vector called *slot_vec* to control occupied time slots. Packets that contain voice information assume to have higher priority than packets that contains ordinary data, and therefore they are allocated before data packets. In addition, voice packets must always use a regular allocation interval.

The slot vector consists of 42 double-slots, which is equivalent to 0.0525 seconds. The time interval is calculated as follows: 42 * 2 * 625 µs (includes both master to slave slot and slave to master slot). The reason why the vector has a length of 42 slots depends on that SCO links must have a special allocation pattern. For the implemented HV3 packet an allocation pattern, which uses every sixth time slot, must be used. Use of HV3 packet allocation pattern results in a 64 kbit/s voice connection. A vector length of 42 slots is very usable since it is a multiple of six.

After such an interval of 42 slots has passed the master allocates the next interval. The active slaves can get different numbers of slots allocated depending on their actual data transmission rates. If SCO links are represented, they always have fixed slots allocated since they are time dependent.

Data packets are then in a random way spread over the vector to get a more real allocation procedure. This procedure is constantly repeated during the entire simulation.

9 Simulations

9.1 Interesting simulation parameters

There are many different parameters, which can be interesting to study when you are going to do simulations of communication networks. Some of the most interesting parameters are: *throughput*, *delay time* and *resource allocation*. These parameters and how they are used in the implementation are described in the following sections.

9.1.1 Throughput

This parameter is a measurement of how the data rate (kbits/s) is varying when data traffic is sent during a simulation. Every slave has a simulation result parameter called *Data transmission rate*, which illustrates how the throughput is changing. The throughput is dependent on the number of participating slaves and their different demands on transmission rate.

9.1.2 Delay time

This parameter illustrates how long time a slave must wait before the connection phase begins. Sometimes the master cannot accept a connection request because of the traffic situation. Delay times are illustrated using simulation result parameters *Data connection delay* and *Voice connection delay* of the slaves.

9.1.3 Resource allocation

This parameter illustrates how must of the piconet channel that is used. The simulation result parameter called *Utilised slots* can be found by the master which controls the channel. The value of this parameter can be between 0 and 1.

9.2 Simulation attributes

Both the master and the slave can be configured before a simulation starts. The model separates the master and the slave from each other and therefore they don’t have the same simulation attributes. Actually, in reality there is no difference between a master and a slave, but due to implementation aspects a separation is made.

The master's simulation attributes include many timers, which are used for a set of different timer lengths. The slave's simulation attributes are more concentrated on the actual slave's traffic characteristics, i.e. how the actual slave shall act. For example, one attribute represents the desired transmission rate.

9.2.1 Master simulation attributes

The following simulation attributes are included in the master process model. A short explanation of the attributes is also given.

- **name** : The name of the master, which can be seen in the project window.
- **model** : The related process model which the master uses.
- **inquiryTO**[7] : Time before the *inquiry timer* expire.
- **master x position** : Determines the master's x-position. This position is fixed during the simulation.
- **master y position** : Determines the master's y-position. This position is fixed during the simulation.
- **my type**[8] : Decides if this module is a master or a slave.
- **pageTO**[9] : Time before the *page timer* expires.
- **parkTO** :Time before the *park timer* expires.
- **print info** : Turns on or off the possibility to get short text messages during a simulation. These messages can only be recognised if the simulation debugger mode is used.
- **sniffTO** : Time before the *sniff timer* expires.
- **unparkTO** : Time before the *unpark timer* expires.
- **unsniffTO** : Time before the *unsniff timer* expires.

[7] The inquiry timer is not used yet. Can be interesting if the model is expanded.
[8] The attribute my type is not used in the implementation, but can be useful later if the master and slave will share the same process model.

[9] The page timer is not used yet. Can be interesting if the model is expanded.

The above-mentioned attributes can either use their default values or be changed by the user. If there are more alternatives these can be seen in a pop-up menu, by clicking on the value field. If the **edit** field, located at the end of the menu list, is used instead, an optional value can be chosen.

The different implemented alternatives, which can be chosen for each respective attribute, are presented in Table 9.1 below. If ARBITRARY is represented the user must type in something. If the field *Change by user* contains a *Yes* the user can type in his/her own value.

Attribute	***Choice***	***Change by user***
Name	ARBITRARY	Yes
Model	blue_master_node_2	No
inquiryTO [slots]	4 slots 8 slots	Yes
master x position [m]	50	Yes
master y position [m]	50	Yes
My type	MASTER SLAVE	No
pageTO [slots]	4 slots 8 slots	Yes
parkTO [slots]	4 slots 8 slots	Yes
print info	Yes No	No
sniffTO [slots]	4 slots 8 slots	Yes
unparkTO [slots]	4 slots 8 slots	Yes
unsniffTO [slots]	4 slots 8 slots	Yes

Table 9.1: Master attributes and defined values.

The different timers can be used to affect the time interval between retransmissions. According to the Bluetooth Specification a usual timer value is 8 slots.

9.2.2 Slave simulation attributes

The following simulation attributes are included in the slave process model. An short explanation about each respective attribute is also given.

- **name** : The name of the slave, which can be seen in the project window.
- **model** : The related process model which the slave uses.
- **active state eff** : Effect value used for calculation of energy consumption in the active mode.
- **parked state eff** : Effect value used for calculation of energy consumption in the parked mode.
- **send packet type** : The packet type the actual slave will use during the simulation.
- **sniff state eff** : Effect value used for calculation of energy consumption in the sniff mode.
- **standby state eff**[10] : Effect value used for calculation of energy consumption in the standby mode.
- **change pos interval** : The time interval between position changes.
- **data block size**[11] : The mean value of data segments size the slave requests.
- **data block variance**[10] : The variance of the data segment size.
- **data rate** : Wanted data rate.
- **data voice request interval**[12] : The mean value of the time interval between a finished transmission and the next transmission. This attribute is used for both voice traffic and data traffic.
- **data voice request interval variance**[11] : The variance of the data voice request interval.
- **link type** : Attribute that decides if the slave will use a ACL link or SCO link.
- **maximum radius from Master** : Restricts the slave's movement.

[10] The **standby** mode is not in use. Can be interesting if the implementation will be expended.
[11] The attributes **data block size** and **data block variance** are used as parameters to a normal distributed function, which generates different data block sizes.

[12] The attributes **data voice request interval** and **data voice request interval variance** are used as parameters to a normal distributed function, which generates different request intervals.

- **movement** : Decides how large moves the actual slave shall be able to do.
- **my type**[13] : Decides if this module is a master or a slave.
- **random data rate** : Turns on or off the function that can generate different data rates for different connections.
- **sniff interval** : Makes it possible to let the actual slave to use sniff mode during the simulation.
- **start mode** : The mode which the slave has in the beginning of the simulation.
- **voice conversation length**[14] : The mean value of the slave's voice connections.
- **voice conversation length variance**[13] : The variance of the voice connection.
- **slave x position** : Determines the slave's x-position when the simulation starts.
- **slave y position** : Determines the slave's y-position when the simulation starts.

The above-mentioned attributes can either use their default values or be changed by the user. If there are more alternatives these can be seen in a pop-up menu, by clicking on value field. If the **edit** field, which is located at the end of the menu list, is used instead an optional value can be chosen.

The different implemented alternatives, which can be chosen for respective attribute, are presented in Table 9.2 below. If ARBITRARY is represented the user must type in something. If the field *Change by user* contains a *Yes* the user can type in an own value.

Attribute	***Choice***	***Change by user***
Name	ARBITRARY	Yes
Model	blue_slave_node_2	No
active state eff [W]	90 mW (Max) 27.6 mW (9600 bps)	Yes

[13] The attribute **my type** is not used in the implementation, but can be useful later if the master and slave will share the same process model.

[14] The attributes **voice conversation length** and **voice conversation length variance** are used as parameters to a normal distributed function, which generates different voice connection lengths.

parked state eff [W]	ARBITRARY	Yes
send packet type	DM1 DH1 HV3	No
sniff state eff [W]	ARBITRARY	Yes
standby state eff [W]	ARBITRARY	Yes
change pos interval [s]	1 s 3 s 5 s 10 s	Yes
data block size [bits]	50 kbits 100 kbits 200 kbits 500 kbits 1 Mbits	Yes
data block variance [bits]	30 kbits 50 kbits 100 kbits	Yes
data rate[15]	8,208 kbits/s 28,728 kbits/s 57,456 kbits/s	No
data voice request interval [s]	3.0 s 5.0 s	Yes
data voice request variance [s]	0.3 s 0.5 s	Yes
link type	ACL SCO	No
maximum radius from Master [m]	10 meters	Yes
movement	Small Medium Large	No[16]
my type	MASTER SLAVE	No
random data rate	Yes No	No
sniff interval	No sniff 0.2625 s (5 * 42 slots) 0.525 s (10 * 42 slots) 1.05 s (20 * 42 slots)	Yes[17]
start mode	Park	No
voice conversation length [s]	10 s	Yes

[15] The reason why the data rate value field contains very special values depends on the implementation.
[16] *Small*, *Medium*, and *Large*: pre-defined values in the implementation. Different input values for the random function.
[17] The value must be a multiple of 42 double slots, that is 42 * 2 * 0.000625 s.

	20 s 30 s 1 min 3 min	
voice conversation length variance [s]	3 s 5 s	Yes
slave x position [m]	ARBITRARY	Yes
slave y position [m]	ARBITRARY	Yes

Table 9.2: Slave attributes and defined values.

9.2.3 Disturber simulation attributes

The following simulation attributes are included in the disturber process model. A short explanation about respective attribute is also given.

- **name** : The name of the disturber, which can be seen in the project window.
- **model** : The related process model which the slave uses.
- **antenna gain** : Decides the antenna gain of the disturber.
- **change pos interval** : Specifies the time interval between the disturber's position updates.
- **jammer x position** : Decides the disturber's x-coordinate when the simulation starts.
- **jammer y position** : Decides the disturber's y-coordinate when the simulation starts.
- **max movement radius** : The maximum distance the disturber can move from the master.
- **movement** : Decides how much the disturber shall be able to change its position.
- **my type** : Classifies the unit to be a disturber.
- **status** : Turns on or off the functionality of disturber.
- **transmitting effect** : Attribute used for affect the disturber's transmission effect.

The above-mentioned attributes can either use their default values or be changed by the user. If there are more alternatives these can be seen in a pop-up menu, by clicking on value field. If the **edit** field, which is located at the end of the menu list, is used instead an optional value can be chosen.

The different implemented alternatives, which can be chosen for respective attribute, are presented in Table 9.3 below. If ARBITRARY is represented the user must type in something. If the field *Change by user* contains a *Yes* the user can type in an own value.

Attribute	***Choice***	***Change by user***
Name	ARBITRARY	Yes
Model	blue_jam_proc	No
antenna gain [dB]	0 dB 1 dB 3 dB 5 dB	Yes
change pos interval [s]	1 s 3 s 5 s 10 s	Yes
jammer x position [m]	10	Yes
jammer y position [m]	10	Yes
max movement radius [m]	20 m 30 m 50 m	Yes
Movement	small medium large	No
my type	JAMMER	No
Status	Active Not active	No
transmitting effect [dBm]	0 dBm 1 dBm 3 dBm 5 dBm	Yes

Table 9.3: Disturber attributes and defined values.

9.3 Simulation result

Before a simulation starts the user can choose which statistical parameters (in OPNET called *object static parameters*) that shall be collected. After that a

simulation has reached its end the simulation results can be viewed with help of OPNET's Analysis configuration tool.

The master, the slave and the disturber has their own result parameters, and these can be studied under section 9.3.1, 9.3.2 and 9.3.3.

9.3.1 Master result parameters

A description of the master result parameters is given below.

- **Lost packets:** This parameter illustrates all packets that have been lost during the simulation. Notice that it is only packets transmitted by a slave and lost during its transmission to the master, which are included. Lost packets in the other direction, i.e. from master to slave, are presented in the slave's result parameters.

- **Utilised slots**: This parameter is being updated with regular intervals to illustrate how the capacity of channel is used. The value can vary between 0 and 1.

- **Transmitted bits to slave** #: This parameter illustrates how many bits have been transmitted to the actual slave. The number of participating slaves is limited to 19, but it can be increased to many more. See Appendix C for more information.

- **Total transmitted bits**: This parameter illustrates how many bits have been transmitted in total, i.e. it is the sum of the above-mentioned parameters (transmitted bits to slave #).

- **Received bits**: This parameter accumulates all by the master correct received bits from the surrounding slaves.

9.3.2 Slave result parameters

A description of the slave result parameters is given below.

- **Active energy**: This parameter shows the energy that has been used during the slave's active time.

- **Active time**: This parameter shows the time the slave has been in active mode.

- **Blocking probability**: This parameter shows the blocking probability for voice connection.

- **Data connection delay**: This parameter shows the connection delay for data transmission requests.
- **Data transmission rate**: This parameter shows how the data transmission rate varies during the simulation.
- **Data transmission rate (expected)**: This parameter shows the expected data rate.
- **Lost packets**: This parameter shows all packets that have been lost during the transmission from the master to the actual slave.
- **Voice connection success**: This parameter shows how many voice requests that have been successfully.
- **Voice connection delay**: This parameter shows the connection delay for voice requests.
- **Voice connection attempts**: This parameter shows the total number of voice connection requests.
- **Sniff energy**: This parameter shows the energy that has been used during the slave's sniff mode time.
- **Parked energy**: This parameter shows the energy that has been used during the slave's park time.
- **Park time**: This parameter shows the time that has been used during the slave's park time.

The parameters that describe the power consumption are calculated with the different state times and state effects. The user can affect the state effects before the simulation starts. The results show the accumulated energy for respective state.

If the total energy is of interest the user must add the different state energies manually.

9.3.3 Disturber result parameter

The only result parameter the disturber offers is **Distance to Master**. This parameter shows how the distance between the disturber and the master has been varying during the simulation.

9.4 How to configure a simulation

Here follows some useful tips for how to configure a simulation.

1. When OPNET is loaded choose *new* from the *File menu*. Then choose *project* from the pull-down menu and click OK. Type in a name for the project and click OK. Then click on the Quit button to abandon the *Startup Wizard*.

2. Push the palette bottom and choose *bluetooth* from the list.

3. Start to place out the master on the working space. **Notice: if the master is placed out after the slave(s) or if the master is cut out and then pasted back on the working space again, this will cause an error when the user tries to start the simulation.**

4. Place out wanted number of slaves. **Notice: the maximum number of slaves is 19.**

5. Place out the disturber on the working space.

6. Configure the master's simulation attributes by right-click on the master and choose *Edit Attributes* from the menu.

7. Configure each respective slave's simulation attributes by right-click on the actual slave and choose *Edit Attributes* from the menu. By shift-clicking on several slaves before the right-click on an arbitrary slave is done, all these slaves can be configured at the same time if the check-box *Apply Changes to Selected Objects* is marked.

8. Configure the disturber's attributes by right-click on the disturber and choose *Edit Attributes* from the menu. Notice that the disturber can be turned on or off with help of the attribute *status*.

9. Choose the master's result parameter by right-click on the master and then click on *Choose Individual Statistics*.

10. Choose the slave's result parameter by right-click on respective slave and then click on *Choose Individual Statistics*.

11. Choose the disturber's result parameter by right-click on the disturber and then click on *Choose Individual Statistics*.

12. Choose *Configure Simulation (Advanced)* from the *Simulation menu*. Right-click on the symbol to be able to set the simulation *Duration* and give the actual scenario (*Name*) and the V*ector file* a name. Then click OK.

13. To start the simulation push the *execute simulation sequence* button.

14. To view the results after the simulation is finished choose *View Results (Advanced)* from the *Results menu*. Search for the actual scenario in the list. Notice that several results can be illustrated in the same diagram if *Statistics Overlaid* is chosen from the left pull-down menu.

To be able to view results from earlier simulations the actual project must first be opened. Then use the same procedure according to point 14.

10 Simulation scenarios

Two different simulation scenarios have been done with the implemented model to be able to verify the model and make some conclusions. Simulation scenario I assumes that the involved slaves only make requests regarding file transfers, i.e. only data traffic. Simulation scenario II assumes that the involved slaves only make requests regarding voice connections, i.e. voice traffic.

10.1 Simulation scenario I: File transfer

10.1.1 Basic conditions

Two different simulations have been done to illustrate how the traffic will be affected depending on the number of slaves. The first simulation includes 6 slaves while the second simulation includes 11 slaves. All slaves use almost the same file size during their transmissions but the expected data rate differs. The request interval by respective slave is arbitrarily chosen. The disturber has been activated during the two first simulations.

The packet losses, which are illustrated in the results below, describe the number of lost packets, which has occurred in direction between all slaves and the master. One additional simulation with the disturber inactivated has also been done.

10.1.2 Simulation parameters configuration

10.1.2.1 Simulation I

Table 10.1 below shows the configuration for Simulation I.

Slave	*Link type*	*Send packet type*	*Data block size (kbits)*	*Data block size var (kbits)*	*Expected data rate (kbits/s)*	*Data request interval (sec)*	*Change position interval (sec)*
Slave_1	ACL	DM1	500	50	8,208	11	10
Slave_2	ACL	DM1	500	50	8,208	12	10
Slave_4	ACL	DM1	500	50	8,208	14	10
Slave_5	ACL	DM1	500	30	28,728	15	10
Slave_3	ACL	DM1	200	30	28,728	21	10
Slave_6	ACL	DM1	500	50	28,728	22	10

Table 10.1: Simulation I: Configuration of simulation parameters.

10.1.2.2 Simulation II

Table 10.2 below shows the configuration for Simulation II.

Slave	*Link type*	*Send packet type*	*Data block size (kbits)*	*Data block size var (kbits)*	*Expected data rate (kbits/s)*	*Data request interval (sec)*	*Change position interval (sec)*
Slave_1	ACL	DM1	500	50	8,208	11	10
Slave_2	ACL	DM1	500	50	8,208	12	10
Slave_4	ACL	DM1	500	50	8,208	14	10
Slave_5	ACL	DM1	500	30	28,728	15	10
Slave_3	ACL	DM1	200	30	28,728	21	10
Slave_6	ACL	DM1	500	50	28,728	22	10
Slave_7	ACL	DM1	500	50	28,728	17	10
Slave_8	ACL	DM1	500	50	8,208	18	10
Slave_9	ACL	DM1	500	100	28,728	19	10
Slave_10	ACL	DM1	500	50	28,728	13	10
Slave_11	ACL	DM1	500	50	8,208	16	10

Table 10.2: Simulation II: Configuration of simulation parameters.

10.1.2.3 Simulation III

The same simulation configuration as in Simulation I is used, but now the disturber is activated too. The idea behind this simulation is to compare the number of lost packets with and without the disturber.

10.1.3 Simulation results

10.1.3.1 Results of Simulation I

The utilisation of the channel during the simulated 5 minutes has a varying behavioural pattern. The mean value is estimated to be about 0.45. The channel is never overloaded. See Figure 10.1 below.

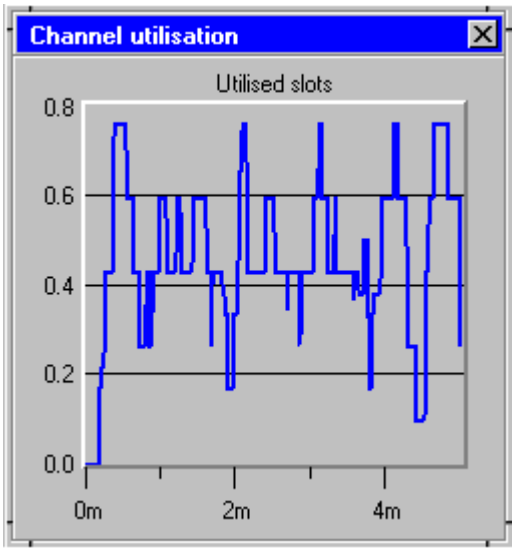

Figure 10.1: Channel utilisation when 6 slaves were involved.

The connection delays are rather small. The maximum delay time for one of the slaves is about 0.05 seconds. This mean that the connection requests are spread out and the number of communicating slaves do not cause any trouble for the traffic. See Figure 10.2 below.

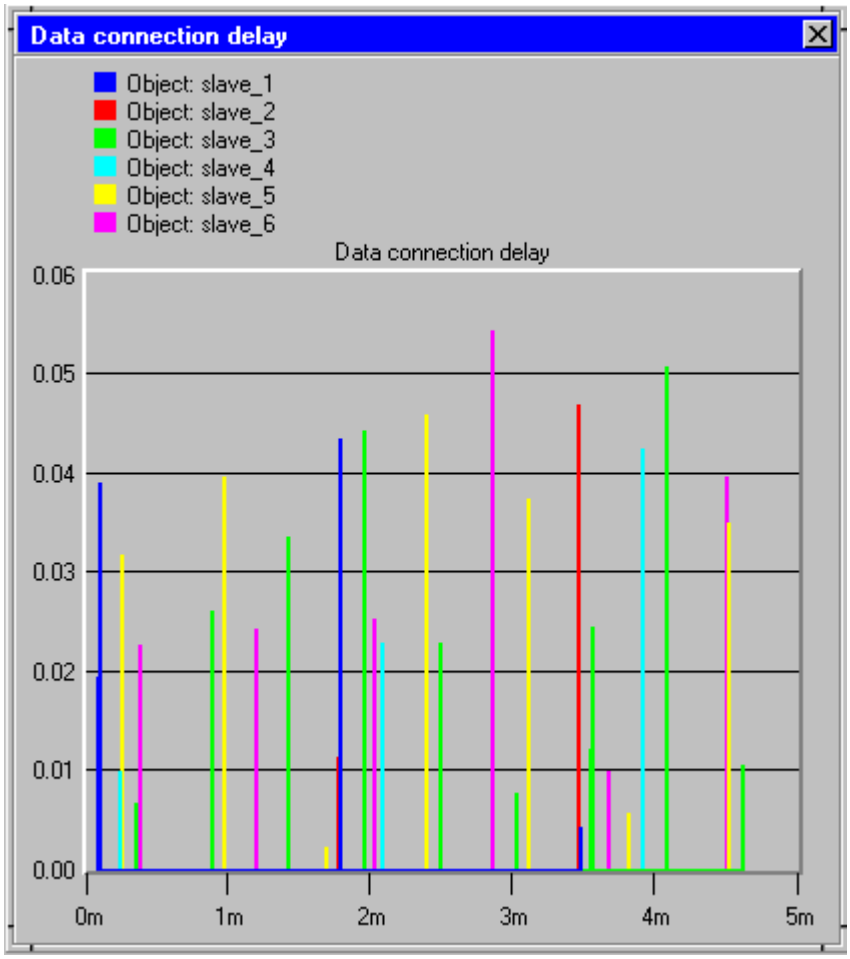

Figure 10.2: Connection delays [s] when 6 slaves were involved.

Since the channel never is overloaded, all slaves will get their expected transmission rates. Figure 10.3 below shows the transmission rate by one of the slaves. The reason why the transmission rate can exceed the maximum level depends on how the calculations are done in the implementation. In reality, the transmission rate shall not be able to exceed the maximum level.

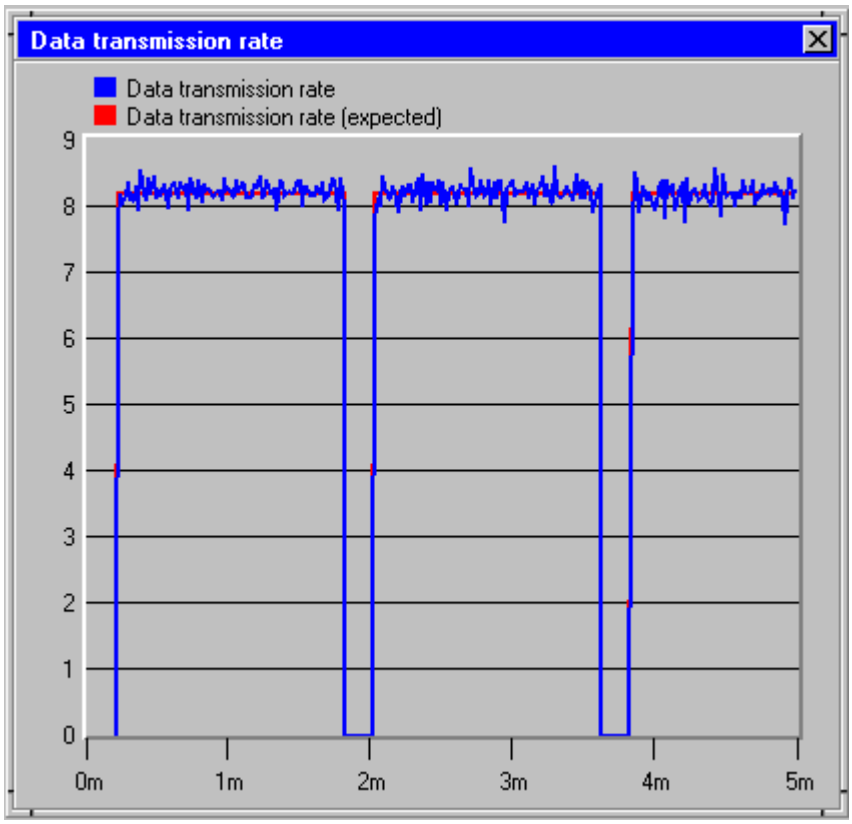

Figure 10.3: Transmission rate [kbits/s] for a arbitrary chosen slave.

10.1.3.2 Results of Simulation II

The utilisation of the channel for the second simulation with 11 slaves involved has a mean value about 0.7. During this simulation the channel has been overloaded once and this happened after about two minutes, see Figure 10.4 below.

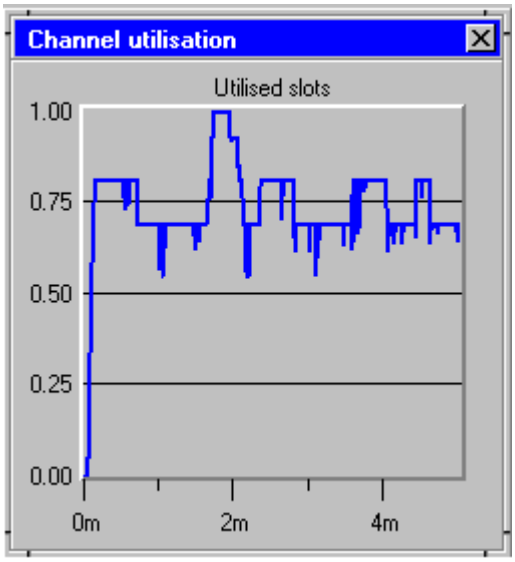

Figure 10.4: Channel utilisation when 11 slaves were involved.

The data connection delays for all of the slaves have different characteristics. The increase of requests have resulted in that the number of communicating slaves most of the time is 7. The consequence of this is that parked slaves have to wait much longer, up to about 50 seconds, before they can get connected. In Figure 10.5 below some of the slaves are represented, not all.

Notice: A slave tries to set up connections until it success.

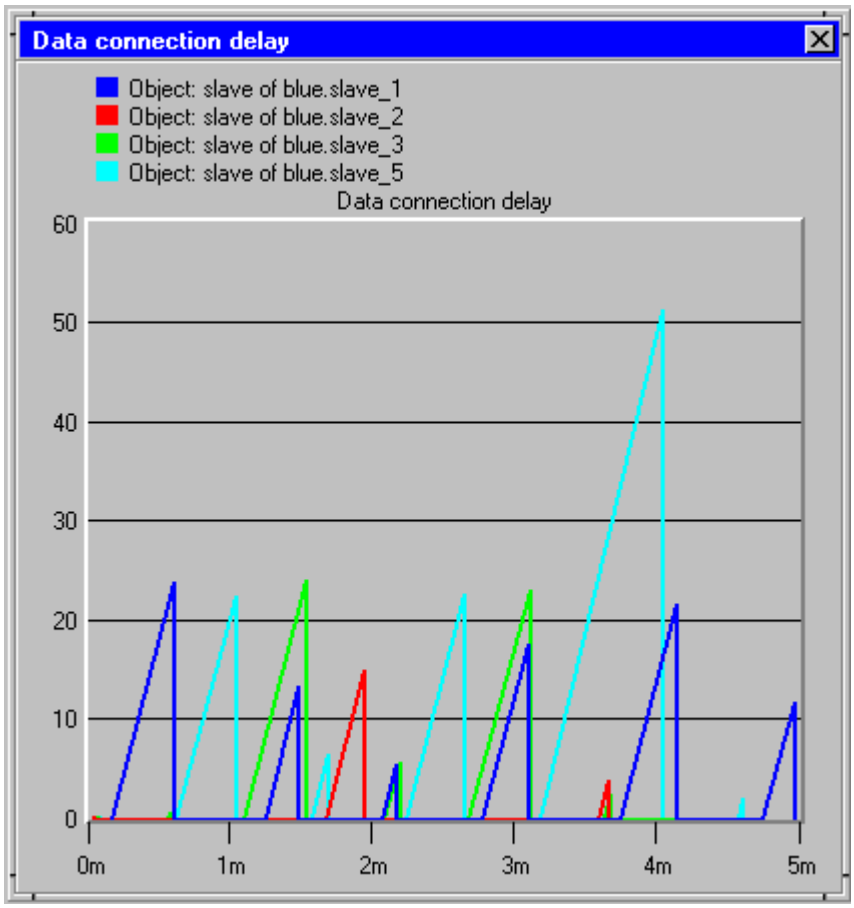

Figure 10.5: Connection delays [s] when 11 slaves were involved.

The data transmission rates of the slaves will now be affected while the channel is used to its maximum. Figure 10.6 below, shows how the real transmission rate by a slave has been decreased a bit. Note, this is the same time as when the channel was overloaded, compare with Figure 10.4.

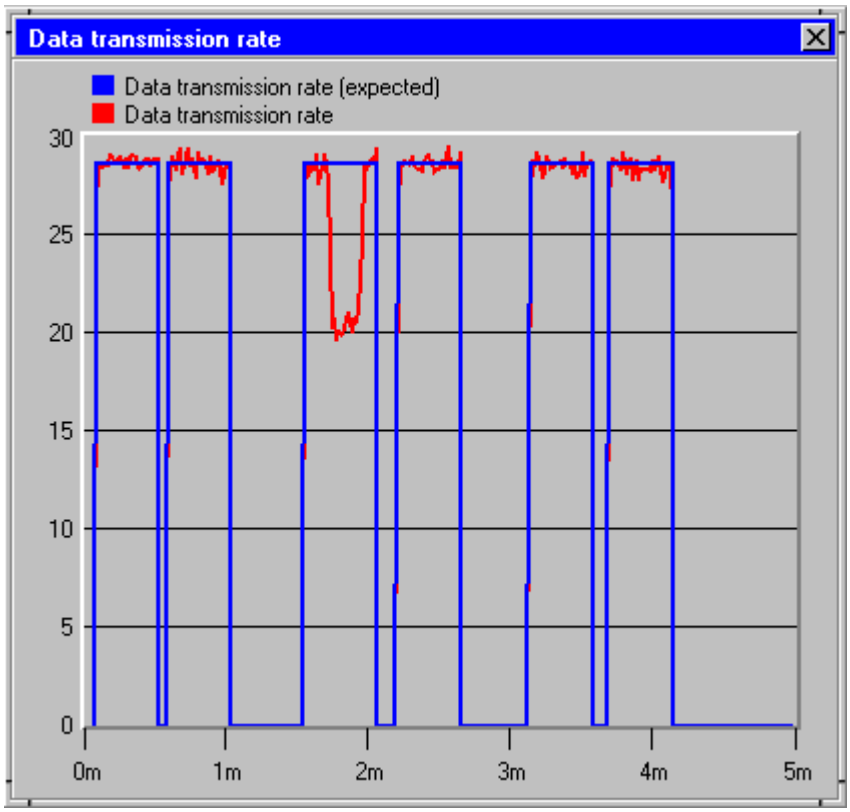

Figure 10.6: Transmission rate [kbits/s] for a slave with decreased transmission rate.

10.1.3.3 Results of Simulation III

Figure 10.7 below shows how the number of lost packet increases when the disturber is activated. About 200 more packets are lost during the 5 simulated minutes.

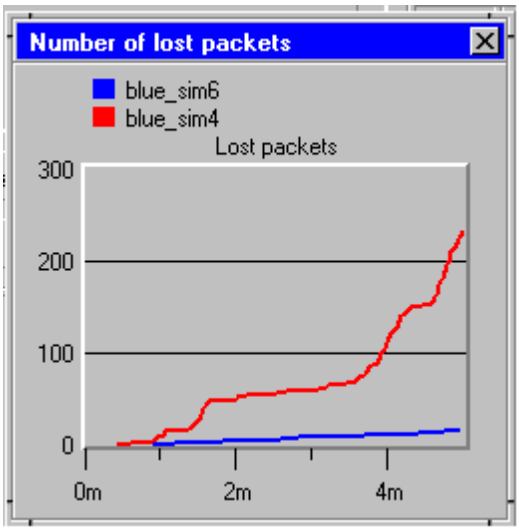

Figure 10.7: Number of lost packets.

Figure 10.8 below shows how the distance between the disturber and the master has been changed during the simulations. The distance during the simulation with 11 slaves is shorter all the time than the simulation with 6 slaves. This has doubtless resulted in more packet losses. Note, also increased traffic within the piconet channel has affected the packet losses.

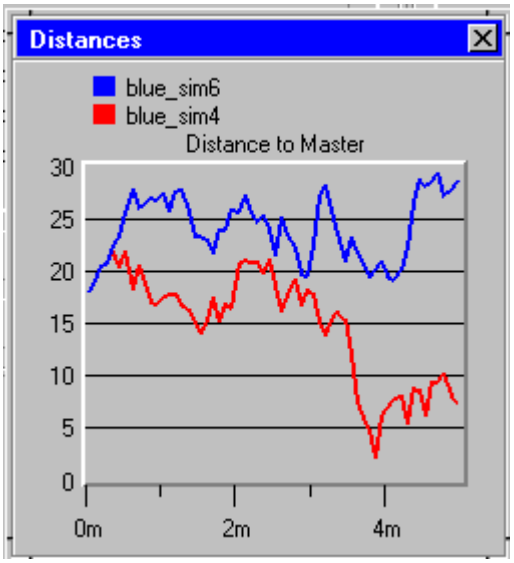

Figure 10.8: Distances [m] between disturber and master.

10.1.4 Conclusions

A piconet with 6 slaves involved as in Simulation scenario I and with the above-assumed characteristics (see Table 10.1) can handle the actual traffic situation without any trouble. Both the connection delays and the transmission rates are not affected negatively.

When the number of slaves increases to 11 with the above-assumed characteristics (see Table 10.2), the throughput of traffic becomes more enhanced. But it was especially the connection delay that shows a sharp increase.

10.2 Simulation scenario II: Intercom telephony

10.2.1 Basic conditions

Four different simulations of a piconet with 7 slaves involved have been done. The length of each simulation is 60 minutes.

The slaves want to get connections with the master to make it possible to communicate over a duplex voice link. The scenario can be thought to be an intercom system with the master as a central unit, which collects all information.

The system will through changes of the connection length by different slaves show different behaviours. The results are illustrated with the following parameters: delay and blocking probability.

10.2.2 Simulation parameters configuration

10.2.2.1 Simulation I

Table 10.9 below shows the configuration for Simulation I.

Slave	*Link type*	*Send packet type*	*Call length (sec)*	*Call length variance (sec)*	*Voice request interval (sec)*	*Change position interval (sec)*
Slave_1	SCO	HV3	20	3	120	10
Slave_2	SCO	HV3	20	3	150	10
Slave_3	SCO	HV3	20	3	180	10
Slave_4	SCO	HV3	20	3	210	10
Slave_5	SCO	HV3	20	3	240	10
Slave_6	SCO	HV3	20	3	270	10
Slave_7	SCO	HV3	20	3	300	10

Table 10.9: Simulation I: Configuration of simulation parameters.

10.2.2.2 Simulation II

The same configuration as in Simulation I but all call lengths are 30 seconds instead of 20 seconds.

10.2.2.3 Simulation III

The same configuration as in Simulation I but all call lengths are 60 seconds instead of 20 seconds.

10.2.3 Simulation results

10.2.3.1 Results of Simulation I

The traffic within the piconet seems to work correct. No blocking has occurred. The delay times seem to be normal and are not affected by any blocking delays. Figure 10.9 shows the behaviour of the connection delays by one of the slaves.

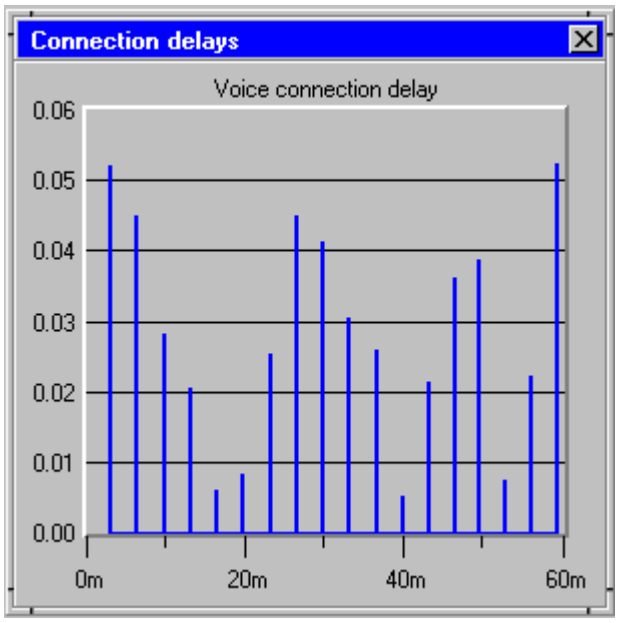

Figure 10.9: Connection delays [s].

Figure 10.10 below illustrates the cumulative distribution of the connection delays. This diagram shows how the probability that the delays are below a certain value increases, if the accepted delay value at the same time increases. For the actual slave, the delays will be shorter than 0.04 seconds with a probability of 90 %.

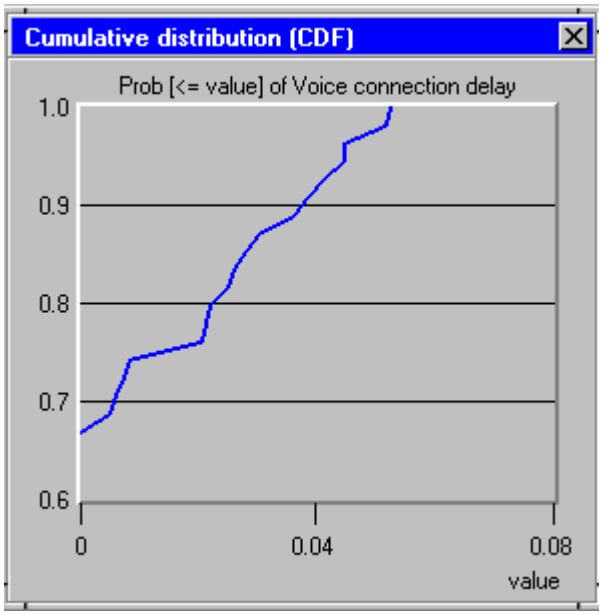

Figure 10.10: Cumulative distribution of the delay times [s].

The reason why no blocking occurs can be explained by the relative short call lengths (20 seconds).

10.2.3.2 Results of Simulation II

The second simulation shows that four of the totally seven slaves are affected by the traffic situation, concerning blocking. Figure 10.11 below shows the average of the blocking probability of respective slave. The values are rather low for all slaves apart from slave number 3 that has a blocking average about 0.06.

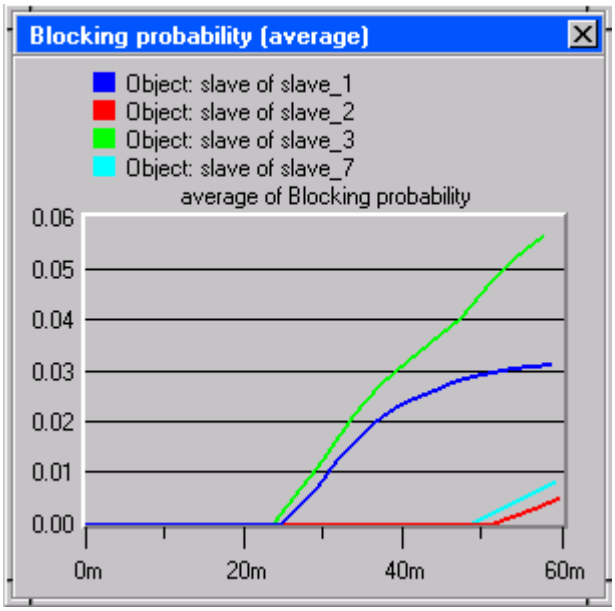

Figure 10.11: Blocking probabilities (average).

When blocking is starting to occur the chances for longer connection delays increases, since more slaves are active at the same time. Figure 10.12 below shows an example of that. Two of the slaves that are involved have connection delays of about 2.5 seconds. Note: the connection delay times are dependent on how the request pattern by the implemented slaves is made.

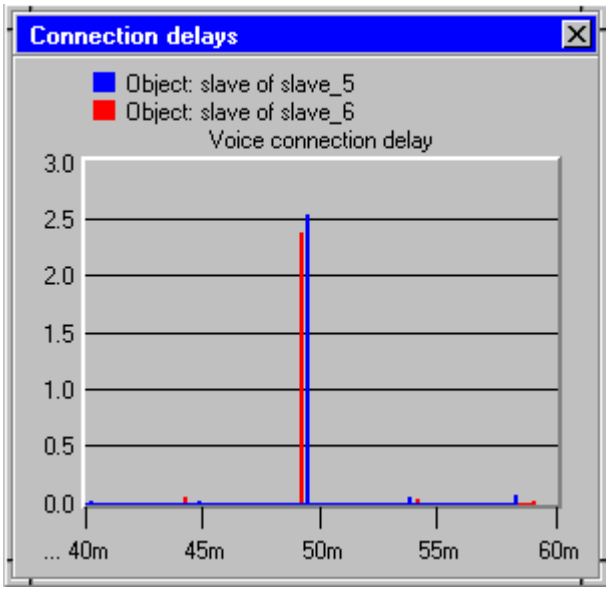

Figure 10.12: Connection delays [s].

10.2.3.3 Results of Simulation III

The third simulation with call lengths of 60 seconds results in more blocking. It is still four of the slaves that are affected, but now the average of the blocking probability is much higher. One of the slaves has an average of about 10 % after the simulated 60 minutes. See Figure 10.13 below.

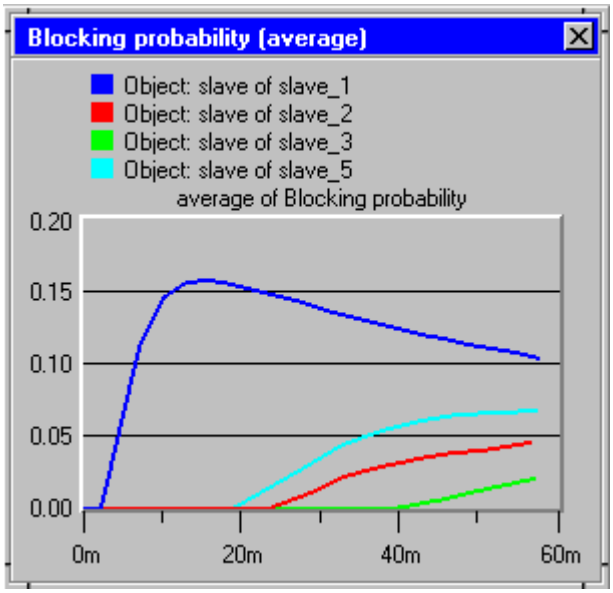

Figure 10.13: Blocking probabilities (average).

These average values of the blocking probability are a bit to high to be acceptable. Normally, blocking values between 0 and 5 percent are okay for telephone systems.

10.2.3.4 Battery lifetime

The following calculation has been done to get an idea about how long time a handheld device can be usable. The scenario can exemplify short-range intern communication (walkie-talkie).

This example takes Simulation I II and III as a starting point for the calculations. The total simulated time is 60 minutes.

The active time and the park time of slave number one have been collected. See Table 10.10 below.

Simulation	*Active time (sec)*	*Active time (per cent)*	*Park time (sec)*	*Park time (per cent)*
1	716.5	0.199	2883.5	0.801
2	680.6	0.189	2919.4	0.811
3	848.1	0.236	2751.9	0.764

Table 10.10: Active and park times.

Assume that the used battery has a capacity of 600mAh and the current consumption for the active and park modes are 10mA respective 0.6mA (see also Table 6.6). These values are increased ten times compared with the values in Table 6.6. The reason for this assumption is that a handheld walkie-talkie will need more current to work. In fact, it is not only the Bluetooth module that will need power supply.

The mean value of the current consumption can now be calculated for respective simulation according to equation 9.1 below.

Mean value of current consumption = *(9.1)*

*(percentage of active time) * 10mA + (percentage of park time) * 0.6mA*

The total battery lifetime can then be decided with equation 9.2 below.

Battery lifetime = 600mAh / (Mean value of current consumption) *(9.2)*

See the results in Table 10.11 below.

Simulation	*Mean current consumption (mA)*	*Battery lifetime (hours)*
1	2.47	243
2	2.38	252
3	2.82	213

Figure 10.11: Mean current consumption and battery lifetime.

For the different simulation scenarios the lifetime of the battery is varying between 9 to 10 days. The lifetime of the walkie-talkie seems to be rather long. In reality other details by the walkie-talkie, maybe need even more current. To be a little more realistic, the battery lifetime will be shorter than 9 days with the above-mentioned conditions.

10.2.4 Conclusions

Systems with relative low call lengths compared with the connection intervals will work without any problems. But if the call length by each user increases, the probability for blocking will also increase according to the results above.

The blocking probability for different scenarios is not only dependent on the respective call length but also on how the different connection intervals are chosen.

The thought was also to do a simulation with the disturber involved to see how the connection delays were affected. But since the results were not so easy to compare and did not give any interesting results, this simulation was skipped. The reason why no interesting results could be found has to do with the wake up sequence by a slave. This delay time are rather long compared with the delay time a single packet loss causes.

11 Proposals for expansions and changes of the existing model

Here follows some proposals on how the existing model can be expanded.

- Common process model for the master and the slave

  This change feels rather natural as in reality a Bluetooth device can both acts as master and as a slave. It will result in a common process model which has both the master's and the slave's characteristics. This change is necessary if a master-slave switch shall be implemented.

- Master-slave switch

  Implement functions which make it possible to let a slave become the master of the actual piconet. Several functions must be added to be able to copy information of the already existing piconet.

- Scatternet

  To make it possible for slaves to get connected with more than just one piconet at the same time. All units must be expanded with one or more parameters, which can separate the different channels.

- Implementation of further packet types

  If further packet types are added to the existing model the behaviour of the slaves can be more varying and real. See Appendix B.

- More varying disturber

  The disturber can be modified so it transmits with different frequencies. Then the disturber can represent another surrounding Bluetooth device.

- Power consumption

  Increased functionality of the already implemented power consumption part.

.

12 Summary and Conclusions

A rather big part of this work has been dedicated to study and understand the Bluetooth technology, especially the lower levels of the specification. The Baseband specification [1] specifies interesting parts that are relevant to use for an implementation. Routines for different connection types like data and voice links, definition of the packets and much more are defined at this level. On the basis of this protocol level I have made a choice to include the following things in my model:

- Representation of the physical channel
- ACL and SCO links
- Different packet types
- Retransmission schemes
- Different states

Other things that the model also include are:

- Movements of the slaves and the disturber
- Packet losses depending on the actual signal-to-interference ratio

Then, after the scope of the model had become more obvious the implementation phase began. This phase was the most time consuming part of the work and I have spent many hours together with OPNET Modeler.

Below follows experiences of the implementation part:

- Many hours must be spent to learn how the structure of OPNET is built-up and to learn how the functionality of OPNET works. Another important part of OPNET is to get a good overview of the built-in kernel procedures.

- To do an implementation in OPNET C/C++ code can be used. From earlier programming experience and of the experience of this implementation, I just want to say: a programming project takes time!

- OPNET offers lots of possibilities. The possibilities that I have made use of in this implementation are just a small part of what OPNET offers. OPNET is a really powerful simulation tool if you learn to how to use it.

The next phase was to design two different simulation scenarios to be able to see the result of the implementation. The first scenario represented *file transfers* between the master and different number of included slaves. The first simulation included 6 slaves while the second included 11 slaves. If the results from these two simulations are compared with each other the following things can be

summarised. The first simulation did not overload the channel and the connection seemed to be relatively short. The number of involved slaves can explain this result. Every time a slave did a new request the master always had resources to accept the slave.

The second simulation worked also rather well, but now the connection delays increased very much. One of the involved slaves got its wanted transmission rate reduced in connection with that the channel was overloaded. During this simulation the number of active slaves seemed always to be 7, which is a maximum for a piconet and that, can explain the very long connection delays.

The next simulation scenario, *Intercom telephony*, represented a situation where 7 slaves wanted to set up a voice connections with the master. The parameters that affected the simulation results most were the lengths of the different connections and the different connection intervals. The three different connection times were set to 20, 30 and 60 seconds. The results of the different simulations showed that the blocking probabilities for the involved slaves increased when the connection times were longer, which seems to be rather naturally. The connection delays began to increase for the second and third simulation.

An example of how the state times from a simulation can be used to calculate the lifetime for a battery was illustrated with help of the *Intercom telephony* scenario.

In the future many companies will doubtless take benefit by the Bluetooth technology. There are lots of conceivable usage areas where Bluetooth can facilitate things for the user. But Bluetooth also has some restrictions, for example when larger quantities of information shall be sent over a piconet channel. The raw data rate is still 1 Mbit/s, which is not enough for all applications. The simulation results above in this report show examples of some of these restrictions.

Different types of simulations will make it easier for maintenance and expansion of communication networks. Simulations make it possible to save both time and money.

Abbreviations

ACL	Asynchronous Connection-Less
AM_ADDR	Active Member Address
ARQ	Automatic Request Scheme
ATM	Asynchronous Transfer Mode
BD_ADDR	Bluetooth Device Address
CAC	Channel Access Code
CRC	Cyclic Redundancy Check
DAC	Device Access Code
FEC	Forward Error Correction
FH	Frequency hop
GFSK	Gaussian Frequency Shift Keying
IAC	Inquiry Access Code
IP	Internet Protocol
ISM	Industrial Scientific Medical
L2CAP	Logical Link Control and Adaptation Layer Protocol
LAN	Local Area Network
LMP	Link Manager Protocol
OSI	Open Systems Interconnect model
PC	Personal Computer
PDA	Personal Digital Assistant
PM_ADDR	Parked Member Address
ppm	parts per million
PPP	Point-to-Point Protocol
RF	Radio Frequency
SCO	Synchronous Connection-Oriented
SDP	Service Discovery Application Profile
S/I	Signal-to-Interference
SIG	Special Interest Group
TCP	Transport Control Protocol
TDD	Time-Division Duplex
UDP	User Datagram Protocol

Appendix

A. Bluetooth audio

Bluetooth offers two different types of voice coding schemes to let voice communication be possible. The voice coding schemes that is supported on the air interface are:

- 64 kb/s log PCM format (A-law or µ-law)
- 64 kb/s CVSD (Continuous Variable Slope Delta Modulation)

More information about the Bluetooth audio can be found on page 139 in Bluetooth Specification Version 1.0 A.

B. Implementation of new packet types

To be able to add new packet types to the model some addendum must be done. Both the master's and the slave's process model will be affected by these changes, and how to do that is described below.

Changes in the master's process model:

1. The new packet type must be defined in OPNET's Packet Format Editor. Compare with already implemented packet types as *blue_dm1_packet* or *blue_hv3_packet*.

2. In the *XMT*-state (transmission state) of the process model a case for the new packet must be added. A new function for transmission of the new packet can be written or add it in the already existing function *send_packet(slave, packet_type).* A new *master_packet_code* must be choosen so the slave can identify the packet. This code is written in the type field of the packet before it is transmitted. Compare with the other already existing packets.

Changes in the slave's process model:

1. Some addendum must be done in the switch module, which can be found in the *send_back*-state of the process model. The new packet type must get an own *case* so the slave knows what to do with the packet. Compare with already implemented packets.

2. The new packet type must be defined as a *Model Attribute* in the *Interfaces menu*. Choose *send packet type* and click on the *Edit Property* button, to be able to add the new packet.

C. Expansion of number of slaves

If more than totally 19 slaves shall be simulated at the same time, the current model must be modified. The following thing must be changed.

1. The parameter *MAXIMUM_NO_OF_NODES* must be set to the desired number of slaves. This parameter is located in the include file *externvar.c*.

2. *not_active_slave_list* must be initiated with the required size.

3. Parameters related to different interrupts must be expanded. See Header block.

4. Local static handlers must be defined in State variable block.

5. Declaration of the static handler must be done in the master's *init* state. Compare already implemented static handlers.

D. Addition of statistical parameters

To add further statistical parameters to the implementation the following must be done:

1. Define a new *Stat-handler* variable in the SV block.

2. Declare the parameter as a *Local Statics* or *Global Statics* in the *Interface menu*.

3. Register the new variable in the *init* block of the process model with help of OPNET's command *op_stat_reg*.

4. Update the variable with OPNET's command *op_stat_write*.

E. Isotropic antennas

Transmitting antennas, which generate a field that looks the same in all directions, are usually called *isotropic antennas*. That is, the energy flow from the antenna is equal in all directions. See Figure A below.

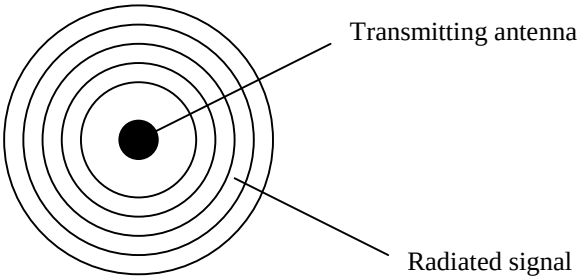

Figure A: Radiation from an isotropic antenna.

F. Profiles

A concise description of the purpose of the different profiles is given in the following sections. Figure B below gives an overview of the relation between the different profiles.

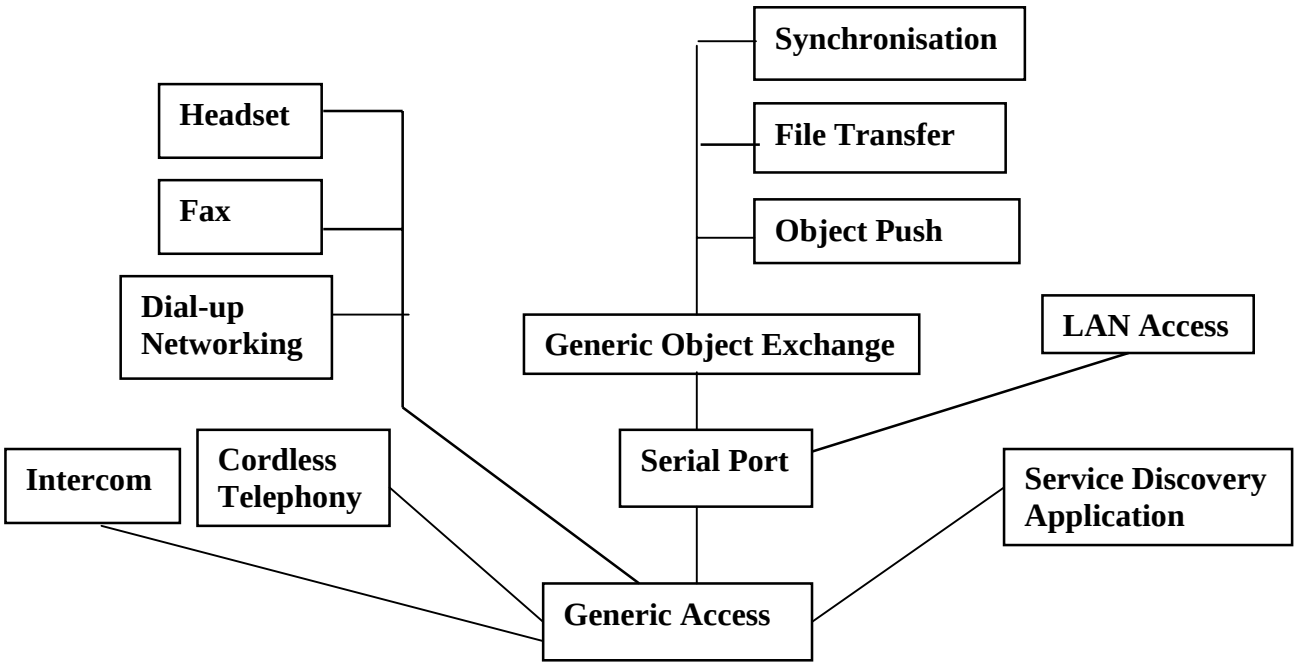

Figure B: Bluetooth profiles.

Generic Access Profile

The purpose of the Generic Access Profile is to introduce common requirements related to modes and access procedures, which will be used by other transport and application profiles. The Generic Access Profile constitutes a base for all other profiles.

A description how devices shall act in standby and connection states to guarantee that link and channel always can be established is an important part. Within the frame of this profile discovery and security procedures are also important.

Other included features are: user interface aspects, coding schemes and more.

Service Discovery Application Profile (SDP)

The Service Discovery Profile defines the protocols and procedures that shall be used by a service discovery application. Such an application is useful if a device wants to locate services in other surrounding Bluetooth devices. The searching for services can be done in several different ways, for example by service class.

Cordless Telephony Profile

The Cordless Telephony Profile defines the protocols and procedures that shall be used by devices that will be able to act like a “Three-in-One phone” (see section 2.2.1) phone.

Intercom Profile

The Intercom Profile defines the protocols and procedures that shall be used by devices implementing the intercom part of the usage model “Three-in-One phone”. It is like a walkie-talkie function.

Serial Port Profile

The Serial Port Profile defines the protocols and procedures that shall be used by devices using Bluetooth for RS232 serial cable emulation.

Headset Profile

The Headset Profile defines the protocols and procedures that shall be used by devices, which will be able to handle headsets. It can for example be a headset that is connected to a mobile phone.

Dial-up Networking Profile

The Dial-up Networking Profile defines the protocols and procedures that shall be used by devices which will be able to connect to an dial-up internet access server. A mobile phone can for example act as a wireless modem for such connections.

Fax Profile

The Fax Profile defines the protocols and procedures that shall be used by devices, which will be able to act like a wireless fax to receive or send, fax messages.

LAN Access Profile

The LAN Access Profile defines the protocols and procedures that shall be used by devices, which will be able to get LAN access using PPP over RFCOMM.

Generic Object Exchange Profile

The Generic Object Exchange Profile defines the protocols and procedures, which shall be used by devices that will have opportunity to exchange different objects. The Synchronisation and File Transfer usage models need this profile.

Object Push Profile

The Object Push Profile defines the protocols and procedures that shall be used by devices, which wants to push an object to another device. The object can for example be an electric business card, which shall be pushed from a mobile phone to another.

File Transfer Profile

The File Transfer Profile defines the protocols and procedures that shall be used by devices, which wants to use the File Transfer usage model. The file transfer can for example be used between two PCs.

Synchronisation Profile

The Synchronisation Profile defines the protocols and procedures that shall be used by devices, which wants to be able to use the Synchronisation usage model (see section 2.2.3). An example, a PC maybe wants to synchronise information with a PDA.

G. OPNET features

If you compare both of the actual tools it feels like OPNET has everything that BONeS has and much more. I have done a dotted list that describes the advantages with OPNET:

- OPNET has a built-in process model that works with transitions. To work in such away gives the person who implements the model advantages. The first advantage is that many protocol specifications are described in a similar manner, i.e. with states and transitions. The second advantage is that the implementation is easy to overview. The third is that another person can get into and understand the model rather quickly.

- OPNET is designed to simulate communication systems and network. It has a very huge model library over already implemented models of for example switches, routers, links and more. That can make a simulation rather easy to configure and that can result in timesavings.

- OPNET offers a very good possibility to study different parameters after a simulation is done. It is simple to choose a specific node or link within the actual network and let the simulation kernel to collect interesting values. It can for example be the throughput of packet or the capacity usage on a link. These collected values can be represented and compared with each other in many different ways with help of OPNET's analysis tool. The results of different simulating scenarios can also be compared.

- OPNET's user interface is nicer and also easier to work with compared with BONeS.

- OPNET can be used both on a UNIX workstation and on Windows NT workstation, which makes OPNET more flexible. The possibilities to work in a more mobile way increases.

- OPNET offers a simple way to let you generate web based result diagram. That is an good advantage if you quickly want to deliver the information to your customers or other interested parties.

- OPNET also offers possibilities to update the tool with other modules.

H. Ericsson Bluetooth Development Kit

The Bluetooth Development Kit is developed to enable early adopters of the technology to quickly accelerate the production of prototype applications. Ericsson has designed the Kit in co-operation with Symbionics.

The Kit provides a flexible design environment, which demonstrates the core features of the Bluetooth technology. Figure C below shows some important parts of the Kit.

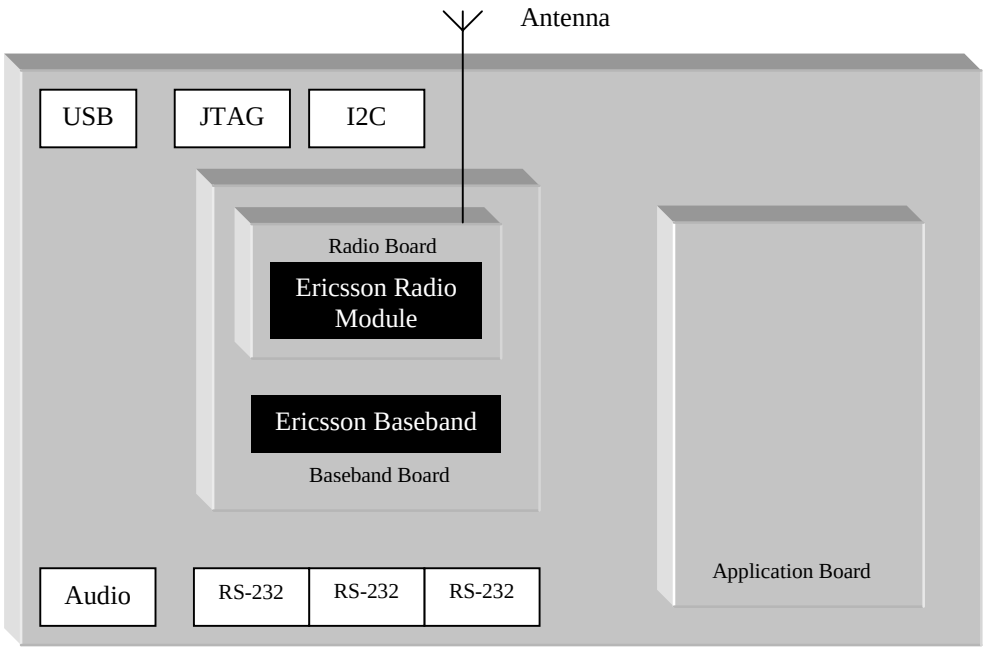

Figure C: The Ericsson Development Kit.

The Kit can be connected to a PC via one of the serial ports (RS-232) or the USB port. Then included software is used to control the board.

The software offers possibilities for the user to set up data- and voice links between two different Kits. Measurements like bit error rate and packet losses can be studied.

References

[1] Bluetooth Specification Version 1.0 A Core

[2] Bluetooth Specification Version 1.0 A Profiles

[3] Bluetooth Protocol Architecture Version 1.0, Riku Mettala, 990825

[4] “Comprehensive Description of the Bluetooth System”, Dan Sönnerstam, 980617

[5] “Kompendium i allmän RADIOTEKNIK”, Olov Carlsson, 850311

[6] “Principles of Wireless Communication”, Lars Ahlin, Jens Zander, 1998

[7] International Edition, "Data and Computer Communications", Fifth edition, p.19-22, Williams Stallings

[8] “Köteori och tillförlitlighetsteori”, Ulf Körner, Studentlitteratur, 1997

[9] OPNET Modeler, Modeling Concepts, Volume 1

[10] OPNET Modeler, Modeling Concepts, Volume 2

[11] Bluetooth Technical Seminar, Kista, 990916

[12] www.bluetooth.net

[13] www.bluetooth.com

[14] bluetooth.ericsson.se/default.asp

[15] http://www.palopt.com.au/bluetooth/

[16] http://bluetooth.ericsson.se/ebc/implementation_kit.asp

[17] http://www.symbionics.co.uk/solutions/bluetooth/Bluetoothkit.shtml