



Budapest University of Technology and Economics  
Faculty of Electrical Engineering and Informatics  
Department of Telecommunications and Telematics

# Packet-Level Simulation of the Bluetooth Physical Layer

Miklós Aurél Rónai

**Master's Thesis**

Advisors:

György Miklós

*M.Sc., Ericsson Research Hungary*

Róbert Szabó

*M.Sc., Budapest University of Technology and Economics*

Budapest, 2001.

# Nyilatkozat

Alulírott Rónai Miklós Aurél, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomatervben csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Budapest, 2001. május

.....  
Rónai Miklós Aurél

# Kivonat

A Bluetooth egy egyszerű és olcsón előállítható frekvencia ugrásos rádiós rendszer. Arra a célra fejlesztették ki, hogy ne legyen szükség az eszközök közötti vezetékekre. Munkámban bemutatom a Bluetooth főbb jellemzőit, ezen belül ismertetem, hogy a technológia hogyan biztosítja a zavarforrások elleni védelmet, illetve hogy hogyan történik a szinkronizáció az egyes eszközök között.

Megvizsgáltam, hogy hogyan lehet egy szimulátorban megvalósítani a Bluetooth technológiát. Alkottam egy szimulátort, mely alkalmas a Bluetooth fizikai rétegének csomag szintű vizsgálatára és amivel az eszközök közötti szinkronizáció pontossága modellezhető. A rendszert kiegészítettem egy véletlen hozzáféréseken alapuló adatkapcsolati réteggel.

Munkámban bemutatom, hogy külső zavarforrás esetén a kommunikáció hatékonysága hogyan csökken. Elemzem a szinkronizáció pontossága és az eszközök áramfogyasztása közötti kapcsolatot. Megmutatom, hogy hogyan lehet csökkenteni a Bluetooth eszközök áramfogyasztását az adatforgalom ismeretében, továbbá elemzem, hogy az áramfogyasztás hogyan függ a vételi ablaktól és a külső interferenciától. Nem folyamatos adatforgalom esetén a vételi ablak mérete erősen befolyásolja az áramfogyasztást, minél nagyobb vételi ablakot alkalmazunk, annál több áramot fogyaszt a készülék. Az áramfogyasztás csökkentése érdekében célszerű olyan kicsi vételi ablakot használni amekkorát csak lehet, továbbá ajánlott a csökkentett áramfogyasztású üzemmódok alkalmazása.

Megvizsgáltam, hogy a sávszélesség hogyan függ az átvitt IP csomagok méretétől és a külső interferenciától. Minél nagyobb IP csomagokat használunk az adatok továbbítására, annál kedvezőbb sávszélesség értékeket kapunk. Megvizsgáltam továbbá azt is, hogy az adatkapcsolati rétegben megvalósított protokoll overhead-je és a backoff-fal eltöltött idő hogyan függ a külső interferenciától. Ha a csomagvesztési arány 0 és 20% között van, akkor a backoff-fal eltöltött idő nem jelentős. Megmutatom, hogy az alkalmazott backoff megoldás  $CV_{min}$  és  $CV_{max}$  paramétereinek különböző értéke jelentősen befolyásolja az elérhető átviteli sebességet.

Az eredmények azt mutatják, hogy a Bluetooth technológia ellenálló a külső zavarforrásokkal szemben, és az energiakímélő üzemmódok alkalmazása nélkül az áramfogyasztás jelentősen megnő. Továbbá az is látható, hogy a szimulátorban alkalmazott véletlen hozzáféréseken alapuló adatkapcsolati réteg jól együtt tud működni a Bluetooth fizikai rétegével. Ezen felül a szimulátor alkalmas a különböző Bluetooth-t továbbfejlesztő megoldások elemzésére.

# Abstract

Bluetooth is a simple, low cost frequency hopping radio system. It was developed to eliminate cables between devices. In my work I introduce the Bluetooth technology. I show how Bluetooth provides protection against external interference, and how it maintains the synchronization between nodes.

I discuss how to implement Bluetooth in a simulator. I have created a simulator which is capable of investigating the Bluetooth physical layer on packet-level. For the investigations I have created a simplified link layer which is based on a random access method.

I show how the external interference decreases the efficiency of the communication between nodes. I have made a numerical analysis on how the power consumption of Bluetooth nodes depends on the accuracy of synchronization and how it can be decreased by changing the receive window size in the knowledge of the traffic pattern. I show how the power consumption depends on the receive window size and on the external interference. In case of continuous traffic, the power consumption is not influenced significantly by the size of the receive window. In case of non-continuous traffic the power consumption highly depends on the receive window size, the longer the receive window the more power is consumed. To decrease the power consumption it is practical to use as short receive window as possible and the usage of the power-saving modes is recommended.

I analyze how the throughput depends on the IP packet size and on the external interference. The larger the transmitted IP packets are the better throughput values we get. I have investigated how the overhead of the implemented protocol and the time spent with backoff depends on the external interference. In case of 0-20% packet loss the time spent with backoff is not significant. I show how the  $CV_{min}$  and  $CV_{max}$  parameters of the backoff mechanism influence the throughput.

The results show that the Bluetooth technology is robust against external interference and can be efficient in power consumption. The results also show that the created simplified link layer, which is based on a random access method works well on the Bluetooth physical layer. In addition the simulator is capable of investigating new solutions for Bluetooth.

# Acknowledgements

I would like to thank all my colleagues in Ericsson Traffic Laboratory for their support, especially György Miklós, Csanád Szabó, Márk Félegyházi, András Rácz, Attila Szlovensák and Szilveszter Nadas.

Special thanks are also due to László Piroska, Zoltán Turányi and András Valkó with whom I have worked together in the past two years.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Related Work . . . . .	1
1.2	Structure of the thesis . . . . .	3
<b>2</b>	<b>Bluetooth Technology</b>	<b>4</b>
2.1	Piconet Concept . . . . .	4
2.2	Physical Channel . . . . .	5
2.3	Bluetooth Packets . . . . .	6
2.4	Access Code and Synchronization . . . . .	8
2.5	Establishing Connections . . . . .	9
2.6	Connection State and Power Consumption . . . . .	10
2.7	Scatternet . . . . .	12
2.8	Error Control in the Case of External Interference . . . . .	12
<b>3</b>	<b>Model Description</b>	<b>15</b>
3.1	Simulator Architecture . . . . .	16
3.2	Implementing the Bluetooth Clock . . . . .	17
3.3	Inaccuracy of Clocks . . . . .	17
3.4	Frequency Selection . . . . .	18
3.5	Modeling the Piconets . . . . .	18

3.6	Modeling the External Interference . . . . .	18
3.7	Internal Interference . . . . .	19
3.8	Discussion of Timing in the Bluetooth Physical Layer . . . . .	19
3.8.1	Simulator based on a polling scheme . . . . .	19
3.8.2	Simulator based on a push scheme . . . . .	21
3.9	Implemented Scheme . . . . .	22
3.10	Simplified Link Layer . . . . .	23
3.11	Backoff Procedure . . . . .	25
3.12	Packet Types . . . . .	26
3.13	Modeling the Link Layer . . . . .	27
3.14	Upper Layers . . . . .	30
<b>4</b>	<b>Simulation Based Analysis</b>	<b>31</b>
4.1	Parameters and Network Structures . . . . .	31
4.2	Effect of External Interference . . . . .	34
4.3	Modeling the Accuracy of Synchronization . . . . .	36
4.4	Synchronization and Power Consumption . . . . .	37
4.4.1	Power Consumption in case of Continuous Traffic . . . . .	39
4.4.2	Power Consumption in case of Non-Continuous Traffic . . . . .	42
4.5	Effect of the IP Packet Size . . . . .	45
4.6	Overhead of the Protocol . . . . .	47
4.7	Several $CV_{min}$ and $CV_{max}$ values . . . . .	51
<b>5</b>	<b>Conclusions</b>	<b>52</b>
<b>A</b>	<b>Simulator Specification</b>	<b>56</b>
A.1	Messages between Space and Bluetooth Nodes . . . . .	56

A.2 Packet types . . . . .	59
A.3 Bluetooth Frequency Selection . . . . .	60



# List of Abbreviations

ACK	Acknowledgment
ACL	Asynchronous ConnectionLess
AM_ADDR	Active Member Address
ARQ	Automatic Retransmission reQuest
BC	Backoff Counter
CAC	Channel Access Code
CBR	Constant Bit Rate
CRC	Cyclic Redundancy Check
CTS	Clear To Send
CV	Contention Window
CVSD	Continuous Variable Slope Delta
DAC	Dedicated Access Code
DIAC	Dedicated Inquiry Access Code
FEC	Forward Error Correction
FH	Frequency Hopping
FHC	Frequency Hopping Channel
G-FSK	Gaussian-shaped Frequency Shift Keying
GIAC	General Inquiry Access Code
HEC	Header Error Correction
IAC	Inquiry Access Code
IP	Internet Protocol

IrDA	Infrared Data Association
ISM	Industrial Scientific Medicine
LAN	Local Area Network
MAC	Medium Access Control
NAK	Not Acknowledged
PC	Personal Computer
PM_ADDR	Park Member Address
QoS	Quality of Service
RTS	Request To Send
SCO	Synchronous Connection-Oriented
TDD	Time Division Duplex
WLAN	Wireless Local Area Network

# List of Figures

2.1	Hop frequency selection . . . . .	5
2.2	SCO and ACL links in a piconet . . . . .	6
2.3	Transmission of multi-slot packets . . . . .	7
2.4	Packet format . . . . .	7
2.5	Preamble . . . . .	9
2.6	Trailer . . . . .	9
2.7	Connection-establishment procedures . . . . .	10
2.8	Receive window . . . . .	11
2.9	Scatternet . . . . .	13
2.10	Automatic retransmission request scheme . . . . .	14
2.11	1/3 FEC to protect the header . . . . .	14
3.1	Simulation architecture . . . . .	15
3.2	Simulator architecture . . . . .	16
3.3	Space behavior in case of the polling scheme . . . . .	20
3.4	Radio range check . . . . .	21
3.5	Overlapping packets . . . . .	22
3.6	Space behavior using the push scheme . . . . .	22
3.7	RTS-CTS Handshaking mechanism . . . . .	24

3.8	Backoff . . . . .	25
3.9	Segmenting an IP packet to Bluetooth data packets . . . . .	27
3.10	Receive window checking . . . . .	28
3.11	Transmitting the IP packet over the Bluetooth physical channel . . . . .	29
3.12	Network architecture with a constant bit-rate traffic generator . . . . .	30
4.1	Network structure with one pair of nodes . . . . .	32
4.2	Network structure with four pair of nodes . . . . .	33
4.3	Throughput in case of one and four pair of nodes transmitting 50000 IP packets with the length of 40 bytes . . . . .	34
4.4	Throughput in case of one and four pair of nodes transmitting 3473 IP packets with the length of 576 bytes . . . . .	35
4.5	Throughput in case of one and four pair of nodes transmitting 1334 IP packets with the length of 1500 bytes . . . . .	35
4.6	Receive window size in function of the intensity of incoming packets . . . . .	38
4.7	Continuous traffic . . . . .	39
4.8	Time spent with receiving and sending compared to the full time of packet transmissions, one pair of nodes, 1334 IP packets with length of 1500 bytes, $CV_{min}=1$ , $CV_{max}=16$ , 0% packet loss, continuous traffic . . . . .	40
4.9	Time spent with receiving and sending compared to the full time of packet transmissions, one pair of nodes, 1334 IP packets with length of 1500 bytes, $CV_{min}=1$ , $CV_{max}=16$ , 10% packet loss, continuous traffic . . . . .	40
4.10	Time spent with receiving and sending compared to the full time of packet transmissions, one pair of nodes, 1334 IP packets with length of 1500 bytes, $CV_{min}=1$ , $CV_{max}=16$ , 20% packet loss, continuous traffic . . . . .	41
4.11	Time spent with receiving and sending compared to the full time of packet transmissions, one pair of nodes, 1334 IP packets with length of 1500 bytes, $CV_{min}=1$ , $CV_{max}=16$ , 10 $\mu s$ receive window length, continuous traffic . . . . .	42
4.12	Non-continuous traffic . . . . .	42
4.13	Time spent with receiving and sending compared to the time of transmitting the IP packets, one pair of nodes, 1334 IP packets with length of 1500 bytes, $CV_{min}=1$ , $CV_{max}=16$ , 0% packet loss, non-continuous traffic . . . . .	44

4.14	Time spent with receiving and sending compared to the time of transmitting the IP packets, one pair of nodes, 1334 IP packets with length of 1500 bytes, $CV_{min}=1$ , $CV_{max}=16$ , 10% packet loss, non-continuous traffic . . . . .	44
4.15	Time spent with receiving and sending compared to the time of transmitting the IP packets, one pair of nodes, 1334 IP packets with length of 1500 bytes, $CV_{min}=1$ , $CV_{max}=16$ , 20% packet loss, non-continuous traffic . . . . .	45
4.16	Throughput in function of packet size, one and four pair of nodes, 10% packet loss, $CV_{min}=1$ , $CV_{max}=16$ , packet size in bytes . . . . .	46
4.17	Protocol overhead transmitting 50000 IP packets with 40 bytes of length, one pair of nodes, $CV_{min}=1$ , $CV_{max}=16$ . . . . .	47
4.18	Protocol overhead transmitting 3473 IP packets with 576 bytes of length, one pair of nodes, $CV_{min}=1$ , $CV_{max}=16$ . . . . .	48
4.19	Protocol overhead transmitting 1334 IP packets with 1500 bytes of length, one pair of nodes, $CV_{min}=1$ , $CV_{max}=16$ . . . . .	48
4.20	Overhead in case of four pair of nodes using 50000 IP packets with 40 bytes of length . . . . .	49
4.21	Overhead in case of four pair of nodes using 3473 IP packets with 576 bytes of length . . . . .	50
4.22	Overhead in case of four pair of nodes using IP packets with 1500 bytes of length . . . . .	50
4.23	Throughput in function of the packet loss, different $CV_{min}$ and $CV_{max}$ values	51
A.1	Bluetooth frequency selection kernel for 79-hop systems . . . . .	60

# List of Tables

2.1	Radio parameters . . . . .	5
2.2	Times establishing connections . . . . .	10
3.1	Data packets used in the simulator . . . . .	26
3.2	RTS, CTS packets used in the simulator . . . . .	26
3.3	ACK packet used in the simulator . . . . .	27
4.1	Slots needed to transmit 2000000 bytes using 40 and 500 byte long IP packets in the case of zero packet loss . . . . .	46
A.1	Data packets used in the simulator . . . . .	59
A.2	RTS, CTS packets used in the simulator . . . . .	59
A.3	ACK packets used in the simulator . . . . .	59
A.4	Frequency selection kernel parameters . . . . .	60

# Chapter 1

## Introduction

The development of wireless technologies makes possible to eliminate cables between devices. This makes the use of devices very flexible, even though the usage of cables gives higher transmission rates than wireless applications.

### 1.1 Related Work

In the following I give an overview about related works in the literature and I introduce the structure of my thesis. A number of short range wireless technologies have been developed and standardized in recent years. The infrared technology of the Infrared Data Association (IrDA) [7] is a low cost, cross platform solution. The connections are one to one, not like in a local area network (LAN), where many to many connections are used. The transmission range is very short, typically zero to one meter and the devices have to be in line of sight. This point to point solution does not support networking.

There are several radio based solutions, such as Bluetooth [1] and IEEE 802.11 [5] with radio based physical layer. The radio waves need no line of sight and the transmission range is longer than by infrared technologies. These technologies support some extent of networking.

In Bluetooth multiple channels are available for communication. The nodes can form an ad hoc network which is called scatternet in Bluetooth. The problem which arises when using multiple channels for communications is that we need to select subgroups of nodes that share a common channel and form a so called piconet and we need to select nodes that act as relays and forward traffic from one channel to another [13].

Several papers deal with this scatternet formation problem. In [11] the authors study the performance implications of forming scatternets from piconets. In [10] András RÁCZ et al. propose the Pseudo-Random Coordinated Scatternet Scheduling (PCSS) algorithm to perform the scheduling of both intra and inter-piconet communications. In this algorithm Bluetooth nodes assign checkpoints which serve as regular meeting points for neighboring nodes when they can exchange packets. The authors compare their solution to an ideal and a greedy algorithm. They found that PCSS achieves higher throughput than the greedy algorithm and the throughput can quite closely match the results of the ideal algorithm. At the same time PCSS consumes approximately the same amount of power as the ideal scheduler, and this is significantly less than the power consumption of the greedy algorithm.

In [12] the authors discuss the cross-layer optimizations in scatternets. This allows links between nodes to be kept active only when absolutely required and scatternet-wide floods to be minimized by caching service discovery results at all intermediate nodes.

The Mobile Ad-hoc Networks (MANET) [8] which is a working group of the Internet Engineering Task Force (IETF) [9] focuses on the routing in ad-hoc networks. The routers are free to move randomly and organize themselves arbitrarily, thus the wireless topology of the network may change rapidly and unpredictably.

In [4] the authors present simulation results of the radio network performance analysis of Bluetooth. According to the authors it is advisable to use long uncoded packet types for transmissions since they allow the largest capacity and therefore they generate the least interference power.



## 1.2 Structure of the thesis

In my thesis I investigate the Bluetooth physical layer on packet-level. I analyze the effect of external interference, the accuracy of synchronization and the power consumption of nodes. I introduce a simplified link layer which is based on a random access scheme. I investigate the overhead of the created protocol and the effect of the IP packet size.

In Chapter 2 I describe the main features of Bluetooth technology. I show how Bluetooth provides protection against external interference and how the synchronization between nodes is maintained.

In Chapter 3 I discuss how the Bluetooth physical channel can be modeled on packet-level. I introduce the implemented link layer and describe the developed simulation tool.

In Chapter 4 I analyze the decrease in efficiency of the communication when interference is present. I analyze the relationship between the accuracy of synchronization and power consumption. I show how we can decrease the power consumption of nodes. I discuss how the IP packet size influences the throughput and investigate the overhead of the protocol.

In Chapter 5 I conclude the thesis.

## Chapter 2

# Bluetooth Technology

Bluetooth is a low cost low power frequency hopping (FH) spread spectrum radio system operating in the industrial-scientific-medical (ISM) band at 2.45 GHz [1] [2] [3]. It was designed to eliminate cables and connectors between cordless or mobile phones, modems, headsets, computers, printers, local area network access points, personal computers (PC), mice, keyboards and so forth.

### 2.1 Piconet Concept

Bluetooth nodes that are in range of each other can set up an ad hoc connection forming a so called piconet. For the communication between nodes a frequency hopping radio is used with a hopping rate of 1600 hops/s. Time is divided into 625  $\mu$ s intervals, which are called slots. In each slot different hop frequency is used. Consecutive slots are alternately used for transmission and reception.

Nodes participating in a piconet share the same physical channel. One of the units of a piconet becomes the master of the piconet, all others become slaves. Only one master can exist in a piconet at any time, but there can be up to seven active slave devices. The nodes can change the master and slave roles.

Every unit in the piconet uses the master identity and clock to follow the hopping channel by selecting the appropriate frequency used in the corresponding slot. To generate the master's clock in a slave node, the slave adds an offset to its own native clock (Figure 2.1).

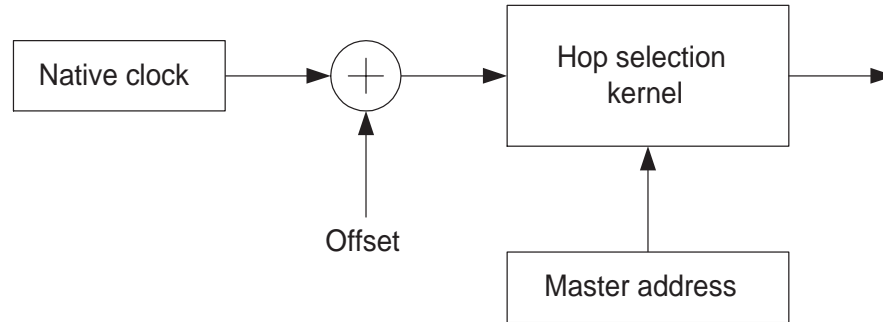


Figure 2.1: Hop frequency selection

A time division duplex (TDD) scheme is used where the master and slave alternately transmit. Master starts transmissions only in even, slave only in odd numbered time slots.

## 2.2 Physical Channel

The specification defines several, equally spaced, 1 MHz wide hops. In countries where the open ISM band is wide, 79 hop carriers have been defined. In countries where the ISM band is narrower (France, Japan and Spain) only 23 carriers have been specified. With Gaussian-shaped frequency shift keying (G-FSK) modulation 1 Mbit/s symbol rate can be achieved. We can see the main parameters of the Bluetooth physical channel in Table 2.1.

Modulation	G-FSK, $h \leq 0.35$
RF band	ISM band, 2.4 GHz
Carrier spacing	1 MHz
RF carriers	23/79
Peak data rate	1 Mbit/s
Peak TX power	$\leq 20$ dBm

Table 2.1: Radio parameters

On a channel two types of link have been defined: a synchronous connection-oriented

(SCO) link typically for voice traffic and an asynchronous connectionless (ACL) link typically for bursty data transmissions. The SCO links are pre-allocated, while ACL connections are controlled by the master unit using a polling mechanism. Communication is enabled only between master and slave devices (Figure 2.2 [3]).

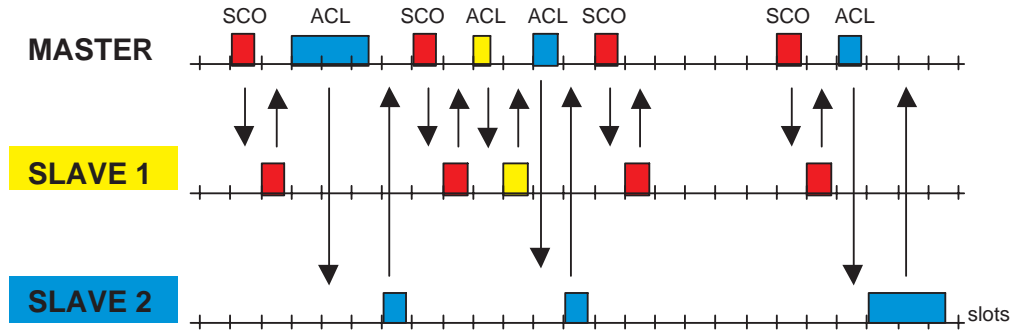


Figure 2.2: SCO and ACL links in a piconet

## 2.3 Bluetooth Packets

The length of packets exchanged between the nodes is one slot, but can be extended up to five slots to support high data rates. Packets are always sent on a single hop carrier, which means that not only one slot packets but also multi-slot packets are transmitted on one hop frequency which is applied in the first slot of the packet. After a multi-slot packet transmission the channel continues the hopping sequence as dictated by the master (Figure 2.3).

Packets have a fixed format as follows (Figure 2.4). They begin with an access code which is derived from the master identity and is unique for the channel. The correlator in the receiver compares incoming signals with the access code stored by the unit. If they are not identical, the received packet is not considered valid on the channel and the rest of the packet is ignored.

The access code is followed by a 54 bit long header. The header contains important control

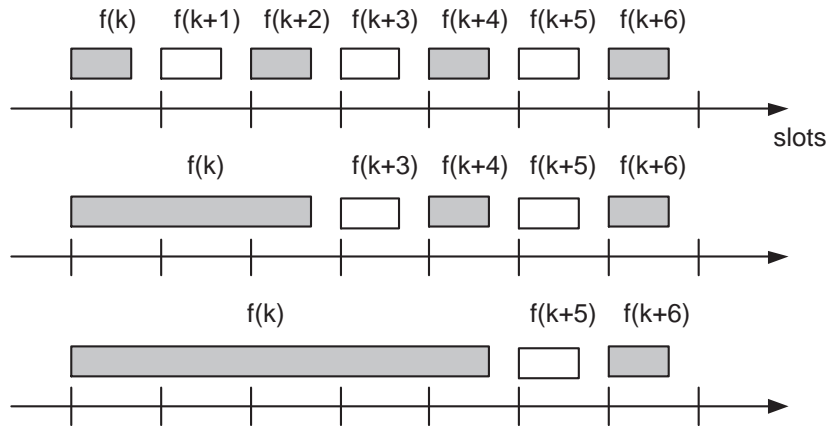


Figure 2.3: Transmission of multi-slot packets

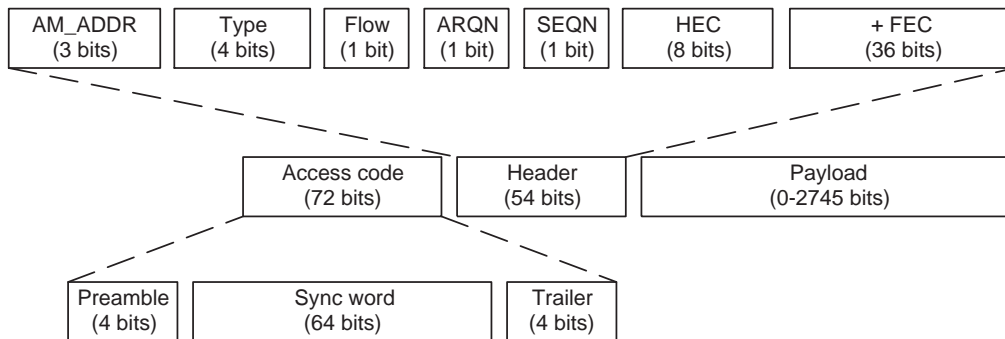


Figure 2.4: Packet format

information, such as an active member address (AM\_ADDR), packet type, flow control bits, bits for the automatic retransmission query (ARQ) scheme and a header error check (HEC) field. The header is encoded with a rate of 1/3 forward error correction (FEC). Payload bits may follow the header.

## 2.4 Access Code and Synchronization

The access code which is very robust and resistant to interference is 72 bits long, if packet header follows it. The access code is used for synchronization, DC offset compensation and identification. All packets sent in the same piconet are preceded by the same channel access code.

In paging and inquiry procedures (Section 2.5) the access code is used as a signaling message and neither header nor payload follows it. In this case the access code is 68 bits long.

Different type of access codes are used in different operating modes. The channel access code (CAC) identifies a piconet, it is included in all packets exchanged in the piconet. The device access code (DAC) is used in special signaling, for example paging and response to paging. The third type of the access code is the inquiry access code (IAC) which has two variations. There is a general inquiry access code (GIAC), which is common to all devices. The GIAC is used to discover other Bluetooth devices in radio range. The dedicated inquiry access code (DIAC) is common for a dedicated group of Bluetooth devices that share common characteristics. The DIAC is used to discover only these dedicated units.

The access code is made up of a preamble, a sync word and possibly a trailer. The preamble consists of four symbols with a fixed 1-0 pattern, either 1010 or 0101, depending on the first bit of the following sync word (Figure 2.5). It is used for DC compensation.

The 65 bit long sync word is derived from the units identity. For the CAC the master's identity is used, for the GIAC and the DIAC dedicated addresses are used, for the DAC the paged units identity is used. The good autocorrelation properties of the sync word

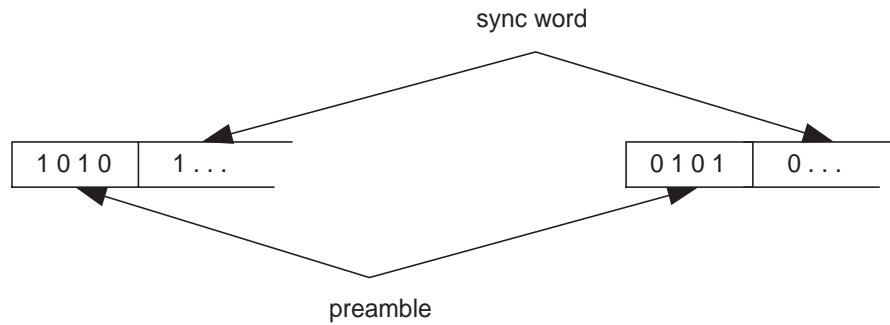


Figure 2.5: Preamble

improve on the timing synchronization process.

The trailer is only present if packet header follows the access code. It is similar to the preamble, it also consists of four symbols, either 1010 or 0101, depending on the last bit of the sync word (Figure 2.6).

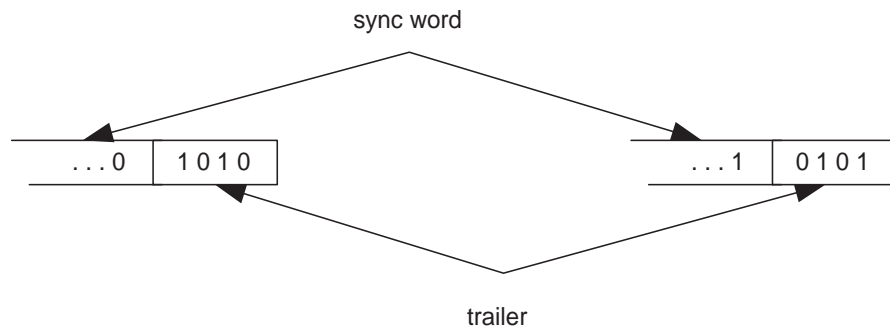


Figure 2.6: Trailer

## 2.5 Establishing Connections

To manage connections two methods are defined, the inquiry which is optional and the page procedure which is mandatory [3]. With the inquiry procedure the unit discovers its neighbors in radio range. During inquiry the node sends IAC signals through the radio interface. Other Bluetooth devices that want to be discovered and receive this signal send

an inquiry response with their own identity.

If the node already knows the destination's identity, the page procedure is used to establish connection. The nodes use dedicated frequencies to communicate in page mode. The unit identity determines these frequencies, which are a subset of the total set of carriers. The nodes choose one of these dedicated carriers pseudo-randomly according to a pseudo random hopping sequence determined by their clocks. If the receiver node detects its access code (DAC), activates itself and sets up a connection with the other node (Figure 2.7).

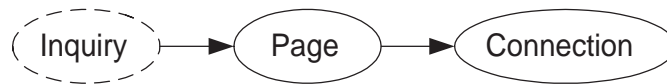


Figure 2.7: Connection-establishment procedures

We can see the typical times associated with establishing connections in Table 2.2.

	Inquiry	Page
Typical time	5.12 s	0.64 s
Maximum time	15.36 s	7.68 s

Table 2.2: Times establishing connections

## 2.6 Connection State and Power Consumption

The master has to send packets to all slaves in connection state periodically to keep them synchronized to the channel. Any packet type can be used for this purpose, since the slaves only need the channel access code to synchronize with. If the slave which is addressed in the poll packet receives the packet, it can respond with any type of Bluetooth packet. In the different operation modes different time periods are used to send the poll packets.

Units can be in four operation modes during the connection state. In active mode the unit actively participates in the piconet. Active slaves listen in the master to slave slots for packets addressed to them. If an active slave is not addressed, it may sleep until the next master transmission. The type indication in the packet header shows the number of slots



the master has reserved for the following transmission. During this time the non-addressed slaves do not have to listen to these slots.

In active mode the receiver is activated  $10 \mu\text{s}$  before the start of the next slot. The receiver searches for channel access code for  $20 \mu\text{s}$  (Figure 2.8). This is called the receive window. If no correct channel access code is received the receiver goes to sleep until the next slot. If a valid access code is received the receiver remains open to receive the rest of the packet.

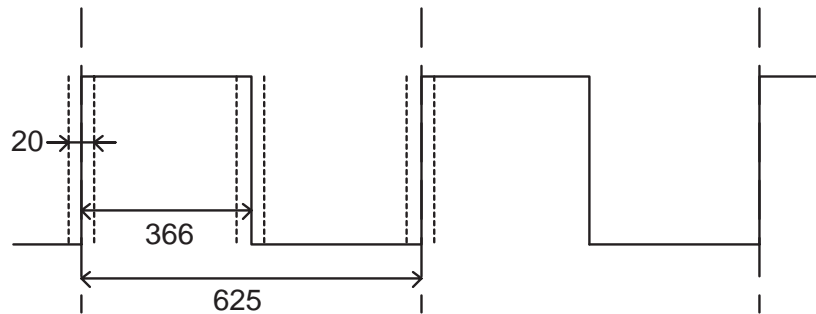


Figure 2.8: Receive window

Furthermore, special power-saving modes are defined to save battery when the traffic is low. In sniff mode the master can start transmission to the slaves only in specified time slots which are called sniff slots and are spaced regularly with an interval of  $T_{sniff}$ . The slave listens for the number of  $D_{sniff}$  slots in every sniff period. If a slave wants to enter in the sniff mode, it asks the master about the value of  $T_{sniff}$  and  $D_{sniff}$ .

In hold mode the unit keeps its active member address (AM\_ADDR), but it will not receive any packets from the piconet. During hold mode the slave can do other things, like scanning, paging, inquiring or attending another piconet. Before entering the hold mode master and slave agree on the time duration the slave remains in the hold mode.

In park mode the slave gives up its AM\_ADDR. It uses two other addresses: the park member address (PM\_ADDR) and the access request address (AR\_ADDR). The PM\_ADDR is used to distinguish the parked devices and the master uses it in the master initiated unpark procedure. The AR\_ADDR is used by the slave in the slave initiated unpark procedure. The parked slave wakes up at regular intervals to listen to the channel in order to

re-synchronize to the master.

The number of units participating in a piconet is limited to 255, but maximum seven slave nodes can be in active, sniff or hold mode. The others are in park mode, where they consume the least amount of energy. The active slave devices can use the sniff and hold modes to deactivate their radio transceivers and save battery power. The less the transceiver is turned on, the less power is consumed.

## 2.7 Scatternet

Several piconets can be created at the same place with overlapping radio areas. The group of overlapping piconets is called scatternet (Figure 2.9 [3]). Every piconet has its own hop sequence, and so the nodes in different piconets can simultaneously transfer data. Because of this the throughput in a scatternet is much greater compared to the case when every node participates in the same piconet.

A node can be slave in several piconets, but it can be master only in one, since the master identity and clock determines the hopping sequence and this must differ in each piconet. Changing piconets the nodes have to select the proper master identity and clock offset to synchronize with the desired piconet. Before changing they inform the master about the time they will not be accessible in the current piconet. A master can also change piconets. In this case the transmissions are stopped in the piconet until the master returns. A multi-hop ad hoc network can built with nodes which are participating in several piconets.

## 2.8 Error Control in the Case of External Interference

The Bluetooth air interface has been designed to deal with external interference, since it operates in the open ISM band, which other devices can also use.

The payload of the packets is protected with an automatic retransmission request (ARQ) scheme. The source transmits and retransmits packets to the destination until receiving a

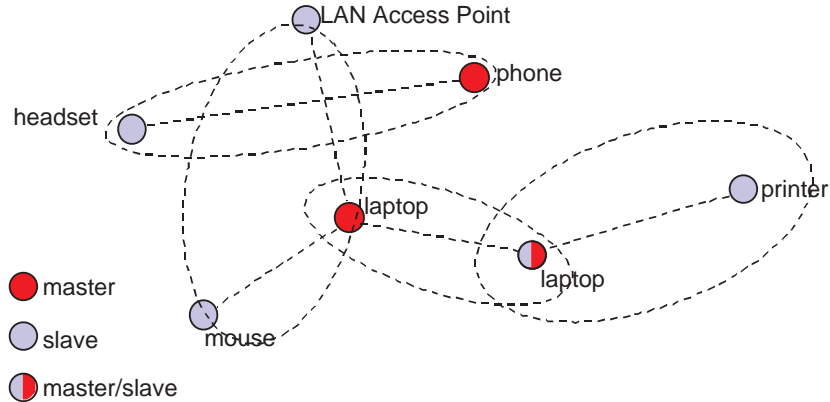


Figure 2.9: Scatternet

positive acknowledgment or timeout expires. The destination receives the packet successfully if the HEC and the cyclic redundancy check (CRC) of the packet are correct. If the reception is successful the destination responds with a positive acknowledgment (ACK), if not with a negative acknowledgment (NAK) packet (Figure 2.10). The slave responds in the first slave to master slot, master responds at the next event it addresses the same slave.

This results in a fast ARQ scheme - only lost packets are retransmitted. The ARQ scheme only works on payloads, which have a CRC code. The packet header and the voice payload are not protected by the ARQ.

Forward error correction (FEC) is also used to protect packets and so reduce the number of retransmissions. Since packet header contains important information about the link, it is always protected by FEC with rate 1/3. For the header a simple 3-times repetition code is implemented (Figure 2.11). FEC with rate 2/3 can be used for the protection of data packets, but in an error free environment it gives unnecessary overhead that reduces throughput [4]. With this FEC scheme each block of 10 information bits is encoded into a 15 bit codeword.

For voice traffic a robust encoding scheme is used, which is based on continuous variable slope delta (CVSD) modulation. This scheme is very resistant to bit errors.

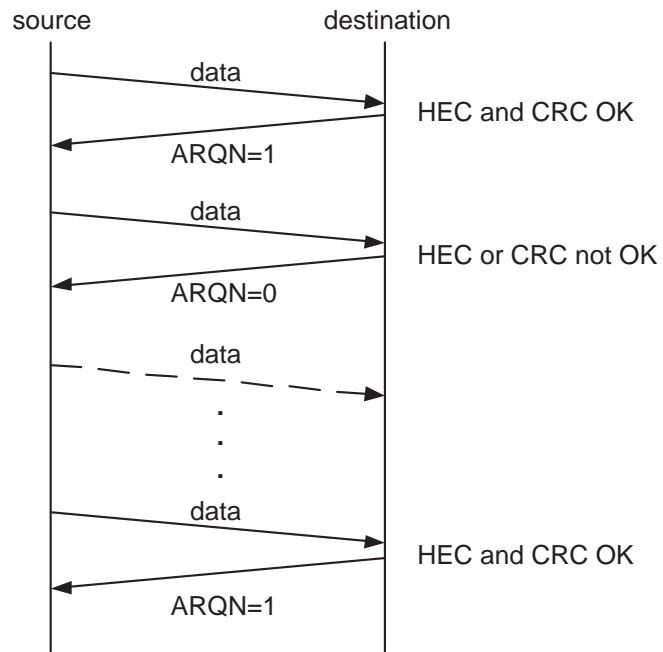


Figure 2.10: Automatic retransmission request scheme

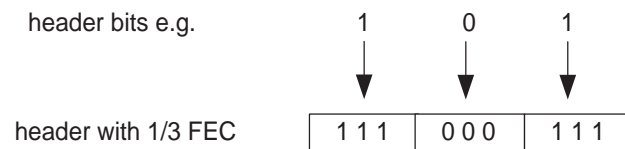


Figure 2.11: 1/3 FEC to protect the header

## Chapter 3

# Model Description

To simulate the Bluetooth physical layer I created a model which is discussed in this chapter. In the following sections I describe how I implemented the Bluetooth clock, the frequency selection, the inaccuracy of clocks, the piconets, packet collisions, the external interference, the timing in the physical layer and the simplified link layer.

For the simulation I used a discrete, event triggered, object oriented simulator environment called Plasma [14]. The implementation of the Bluetooth physical layer is called Space, the implementation of the link layer is called BluetoothLL in the simulator.

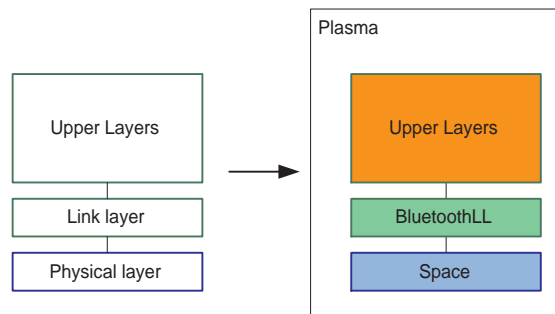


Figure 3.1: Simulation architecture

### 3.1 Simulator Architecture

The architecture of my simulator is seen on Figure 3.2. During upper layer communication IP packets are generated. These IP packets are sent to the link layer, which is called BluetoothLL. The BluetoothLL is responsible for the communication between Bluetooth nodes. It segments and reassembles IP packets, builds up connections between nodes, sends Bluetooth packets through the physical layer, checks whether the transmission is successful.

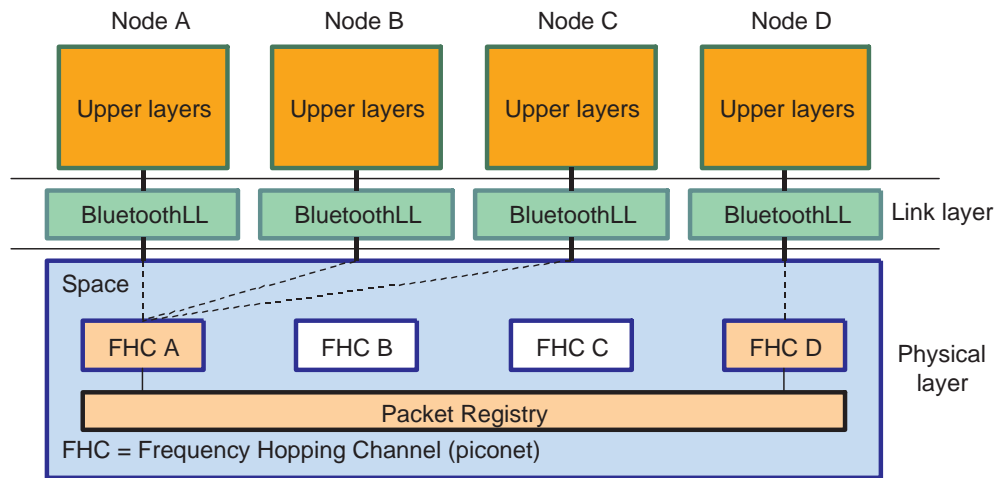


Figure 3.2: Simulator architecture

The BluetoothLL sends the Bluetooth packets to the physical layer, which is called Space. The Space is responsible to check whether the nodes are in radio range, whether they are in the same piconet and whether there is any frequency collision during packet transmissions. The Space also simulates the time spent for transmitting packets and the external interference.

In the simulator the piconet is modeled with the frequency hopping channel (FHC). The Space forwards the packets received from the sender node to the other nodes in the FHC of the sender. Because of the implemented MAC every node in the senders piconet receives every packet from the sender. The BluetoothLL of the receivers checks whether the packet was for the corresponding node and whether it fits in the receive window of the node. If

all the Bluetooth packets of an IP packet are received, the BluetoothLL reassembles the IP packet and sends it to the upper layers.

## 3.2 Implementing the Bluetooth Clock

I had to make several decisions how to model the specification in my simulator. First I had to decide how to implement the Bluetooth clock. I found two possibilities to do this. One solution is to use a counter which shows the number of the current slot. This counter is increased by an event at the beginning of each slot, in normal case at every  $625 \mu\text{s}$ . This solution generates too much unnecessary events in the simulator which would slow down the simulations.

The second solution is that the simulator calculates the number of the current slot only if it is needed. To do this the simulator asks the virtual time from the environment and calculates the number of the current slot dividing the virtual time by the slot length, in normal case by  $625 \mu\text{s}$ . This procedure which determines the number of the current slot is called only when needed, for example when a node wants to send a packet it has to determine the frequency in the current slot and so it has to know which is the number of the current slot. This method does not generate unnecessary events. I have chosen this solution to implement the Bluetooth clock in my simulator.

## 3.3 Inaccuracy of Clocks

To model the inaccuracy of the Bluetooth clock I decided to use a slot length parameter which can be different in each node, in normal case this parameter is  $625 \mu\text{s}$ . The nodes calculate the number of the corresponding slot with this parameter, so they divide the virtual time by the slot length parameter to get the number of the current slot.

### 3.4 Frequency Selection

To implement the frequency selection I saw two possibilities. One way can be that the node chooses a frequency randomly with a uniform distribution. This is very easy to implement. To enable more detailed simulations I decided to use another solution and I implemented the Bluetooth frequency selection kernel in the simulator (Appendix A.3).

### 3.5 Modeling the Piconets

I model the piconet with a structure called frequency hopping channel (FHC). This stores the identity of nodes participating in a piconet. For every piconet there is a corresponding FHC. The number of FHC members is not limited in the simulator. Every unit in the FHC uses the master address and clock to select the appropriate frequency in the corresponding slot. To generate the master's clock in the slave nodes, the slaves add an offset to their native clock.

A node is in one of the following states: either it is tuned to frequency hopping channel of a master, or it is idle. Accordingly, in the simulator a node can be associated with an FHC or can be a stand alone unit. For each node there can be one associated FHC and it is created by the node on demand, when it wants to be master. A node can be associated with only one FHC at a time, since at a given time instant a node can follow only one frequency hopping sequence.

### 3.6 Modeling the External Interference

The external interference is due to other devices operating in the same frequency domain as Bluetooth, for example microwave ovens and wireless local area networks (WLAN). To model the external interference I introduced the packet loss parameter. This parameter can be set in every FHC independently. At every incoming packet the Space chooses a random number uniformly between 0 and 1, and if the number is less than the packet



loss parameter of the FHC, the packet is lost. The damaged packet is also sent to the destinations, so that they are informed about every packet transaction on the physical channel.

### **3.7 Internal Interference**

The internal interference is due to collided packets which are sent on the same frequency at the same time. In the Space I created a packet registry to handle the checking of packets whether they collide or not. Due to internal interference packets can also be lost.

### **3.8 Discussion of Timing in the Bluetooth Physical Layer**

The main thing that Space has to do is to model the time spent for packet transactions. The packets which Space receives from the BluetoothLLs are forwarded to the destinations only if the time is expired which is needed for the transmission of the packets. I investigated two schemes to model the Bluetooth physical layer.

#### **3.8.1 Simulator based on a polling scheme**

First I investigated a polling scheme. This means that at the end of each slot, the Space sends a slot finish message to the BluetoothLLs, tells them what happened in the last slot and asks whether they want to send a packet through the physical channel in the next slot.

The slot finish message describes the status of the channel in the last slot. This can be different at every node, so the nodes receive information about only those events which influence their operation. The message tells the nodes whether there was a transmission on the FHC they participate in, which node was the destination of the packet and whether the transaction was successful or not. The node interprets the slot finish message and

performs all necessary actions according to the packet they received in the slot finish message.

If the node wants to send a Bluetooth packet, it responds to the slot finish message with a send packet message that contains all necessary information about the Bluetooth packet it wants to send on the channel (Appendix A.1).

Figure 3.3 shows how Space works in the polling scheme. At the end of each slot the Space initiates the notification of all nodes about the events in the last slot. The Space checks per node using the packet registry whether there was any frequency collision or packet loss due to external interference. After the events in the last slot are evaluated, the Space sends the slot finish message to every node which were influenced during the last slot. The nodes who want to send packets in the next slot respond with a send packet message. So in the beginning of the slot the Space becomes aware of the transmissions initiated by the nodes. The Space collects all packet transmissions on all the FHCs in a given slot and saves them in the packet registry.

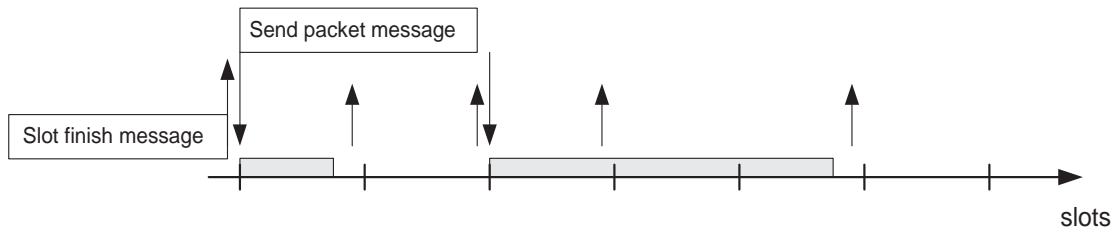


Figure 3.3: Space behavior in case of the polling scheme

Since Space gives the slot time synchronization to the nodes, the packet transmission and reception timing has a resolution of one slot. This scheme introduces an artificial delay: a Bluetooth packet is delivered at the end of the slot even if its reception were finished earlier in reality.

### 3.8.2 Simulator based on a push scheme

I investigated a push scheme to model the Bluetooth physical layer, as well. In this case the Space does not send any polling messages to the nodes. The nodes can send their packets to the Space any time they want to. The Space forwards the packets to the destination after the transmission time elapsed.

After receiving a packet the Space stores it in the packet registry where it remains for the time of six slots. The packets include the length of the packet and the transmission rate. These two parameters tell the Space how much time is needed to transmit the packet on the physical channel. After this time the Space sends the packet to the destinations.

The destinations are those nodes which are associated with the same FHC as the source, and which are in the radio range of the source. The radio range is calculated from a radio range parameter and the x, y positions of the nodes (Figure 3.4).

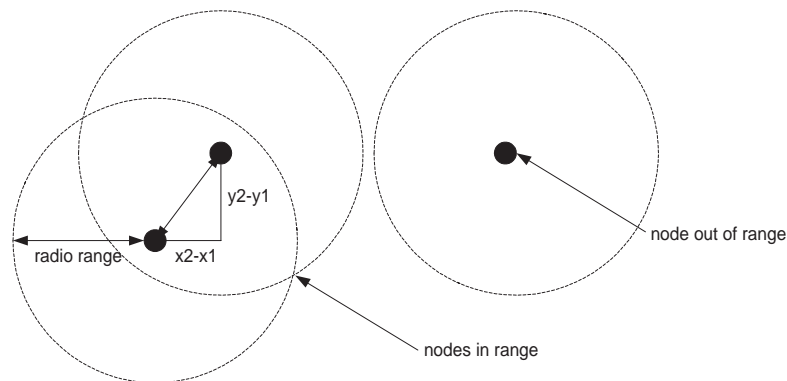


Figure 3.4: Radio range check

The packet registry is used to check whether packets collide in frequency or not. Before sending a packet to the destinations, the Space checks the packet registry whether there are other packets on the same frequency and whether they are overlapping in time (Figure 3.5).

In case of multi-slot packets the Space sends to the destination nodes not only the packet, but sends a message at the end of the header (Figure 3.6). It is needed to tell the destination node which frequency to use for receiving the multi-slot packet in the following slots

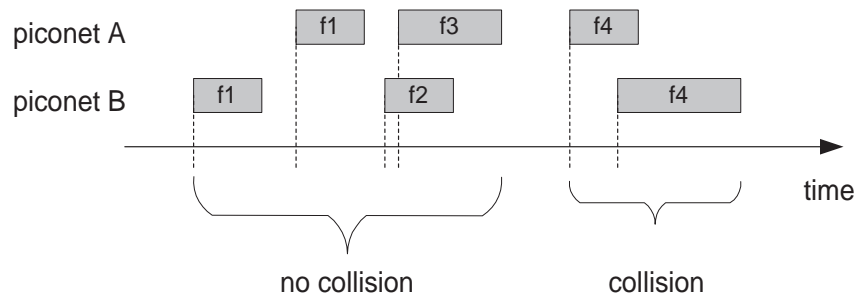


Figure 3.5: Overlapping packets

and not to change the receiver's frequency according to the FHC's frequency sequence. If the header message is transmitted successfully, the receiver of the destination nodes will remain on the same frequency in the following slots. If the transmission of the header message fails (because of external interference or frequency collision), the destination node turns its receiver off and in the next slots it follows the frequency sequence of the FHC.

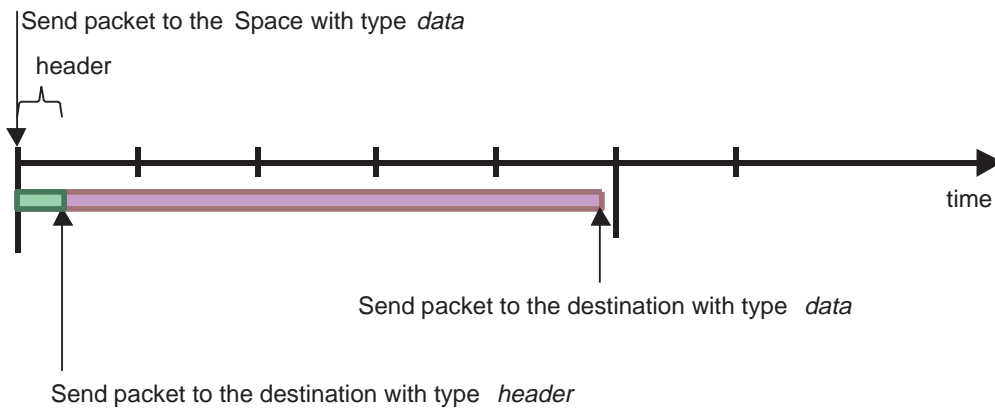


Figure 3.6: Space behavior using the push scheme

### 3.9 Implemented Scheme

In the polling scheme the Space polls all the nodes at the end of each slot with the slot finish message. In this case the Space has to do a lot of things. It has to handle the slot alignment, select the appropriate frequency in every FHC, control the multi-slot transactions, check

the frequency collisions, model the time spent for the packet transmissions and model the external interference. The BluetoothLL only has to answer to the slot finish message and check if the received packet fits its receive window. So this scheme needs a more complex Space object but a simpler BluetoothLL. This scheme is too centralized which may make the simulations slower.

The push scheme models the real life better than the polling scheme. In this case messages are only sent if there was any activity on the channel. In the push scheme the Space checks frequency collisions and models the time spent for packet transmissions and the external interference. The BluetoothLL has to do much more in this case, it has to handle the slot alignment, control multi- slot transactions, select the appropriate frequency and check if the packet fits the node's receive window. So the push scheme needs a simpler Space but a more complex BluetoothLL. This scheme is more distributed than the polling scheme.

I decided to implement the push scheme because it is closer to the reality and may be faster than the polling method.

### 3.10 Simplified Link Layer

I simplified the Bluetooth link layer and applied a random access method for the communication between nodes which is based on the IEEE 802.11 protocol [5]. As an advantage, the random access protocol is easy to implement and yet it provides on demand resource allocation. The potential disadvantages are that it can not handle quality of service (QoS), and in case of high load there are many collisions which decreases radio channel utilization.

The random access MAC in my model is a handshaking mechanism, consisting of request to send (RTS), clear to send (CTS), data and acknowledgment (ACK) packets (Figure 3.7). The source node that wants to send data packets to the destination waits until the current data transfer in the piconet is finished. Then in the following slot it sends a RTS packet to the destination, and waits for a CTS packet. If it receives a valid CTS from the destination in the next slot, it starts to transmit the data packets. If no valid CTS response is received,

the node goes to an exponential backoff wait period and resends its RTS packet at a later, randomly chosen slot. In my model the RTS is resent again and again until a valid CTS response is received.

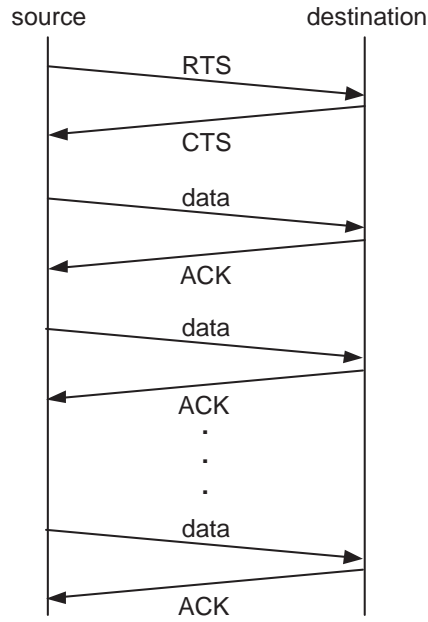


Figure 3.7: RTS-CTS Handshaking mechanism

During the backoff period the node continues to receive packets. If it receives an RTS or CTS packet it waits until the new data transfer in the piconet is over and continues the wait afterwards (Figure 3.8). If it receives an RTS packet addressed to it, it responds with a CTS, and engages in a transmission procedure.

The ARQ scheme in my model works as follows. The source transmits and retransmits packets to the destination until it receives a positive acknowledgment. The destination sends the acknowledgments in ACK packets in the next slot after data packet reception. If the reception was successful the destination responds with a positive ACK, if the reception fails the destination does not respond anything.

By default the receive window is the same as defined by Bluetooth in connection state. The receiver is activated  $10 \mu\text{s}$  before the start of the next slot and remains open for  $20 \mu\text{s}$ . The length of the receive window can be modified during simulations.

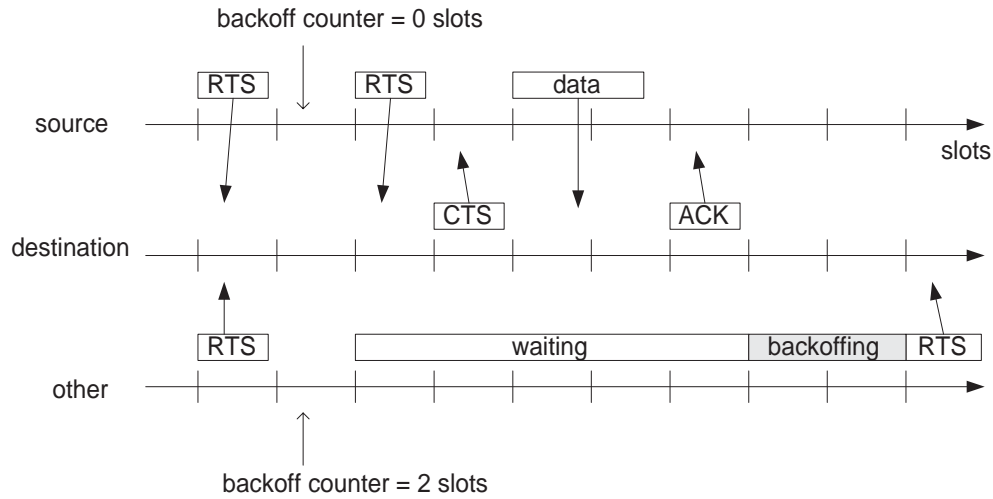


Figure 3.8: Backoff

In the model it is enabled for slave devices to communicate not only with the master but with each other also and nodes can start their transmissions in each slot.

### 3.11 Backoff Procedure

Backoff is used to avoid packet collisions. Packet collision occurs if two or more nodes send an RTS packet in the same slot on the same frequency. If a collision occurs, the nodes whose RTS packets are collided calculate a time until they only listen and do not send anything to the channel.

The backoff works as follows. Every node has a positive integer variable, called backoff counter (BC). The BC is decreased at the end of each slot when there is no active transmission in the piconet. The node sends its RTS packet to the destination, if the value of BC reaches zero.

The value of BC is reset after a successful or unsuccessful RTS-CTS message exchange to a random value chosen uniformly from the interval  $[0, CV]$ , where  $CV$  is the contention window. After a successful RTS-CTS exchange, the value of  $CV$  is set to  $CV_{min}$ . After an unsuccessful RTS-CTS exchange, the value of  $CV$  is doubled. If the value of  $CV$  exceeds

the ceiling  $CV_{max}$ , it is set to  $CV_{max}$ .  $CV_{min}$  and  $CV_{max}$  are constants.

### 3.12 Packet Types

I used the same data packet formats as defined in Bluetooth with a slight modification. Bluetooth data packets are one, three or five slots of length. I extended this by two and four slots long packets. In Table 3.1 the data packet types can be seen. There are one, two, three, four and five slot packets and there are two subtypes of each packet. In one subtype FEC is used with rate  $2/3$  to protect the payload, in the other subtype no FEC is used. In the table we can see the maximum payload of each packet and the total amount of bits in case of the maximum payload. In the table we can also see the guard time in  $\mu s$ . During this period the node's transceiver hops to the next frequency of the sequence.

Slots	FEC	Max payload	Sum bits	Guard time
1	1	136	366	259
1	0	216	366	259
2	1	548	996	254
2	0	838	996	254
3	1	968	1626	249
3	0	1464	1622	253
4	1	1380	2248	252
4	0	2090	2248	252
5	1	1792	2865	260
5	0	2712	2870	255

Table 3.1: Data packets used in the simulator

I created RTS, CTS and ACK packets in my simulator. We can see the properties of the RTS and CTS packets in Table 3.2. The ACK packet is defined in Table 3.3.

Slots	FEC	Max payload	Sum bits	Guard time
1	1	136	366	259
1	0	136	286	339

Table 3.2: RTS, CTS packets used in the simulator



Slots	FEC	Max payload	Sum bits	Guard time
1	1	0	126	499
1	0	0	126	499

Table 3.3: ACK packet used in the simulator

Packets are always sent on a single frequency which is applied in the first slot. After transmitting multi-slot packets the node continues the hopping sequence as dictated by the master address and clock.

### 3.13 Modeling the Link Layer

In the simulator the link layer is called BluetoothLL. The BluetoothLL is responsible for the communication between nodes. If the BluetoothLL receives an IP packet from the upper layers, it first segments the IP packet into Bluetooth data packets (Table 3.1). Segmentation of an IP packet means that as many data packets are generated as needed to transmit the IP packet. For example if an incoming IP packets size is 1500 bytes (12000 bits), the BluetoothLL generates five data packets if no FEC is used. Four of them are five, one of them is four slots long (Figure 3.9). The BluetoothLL puts these packets in a queue and creates an RTS packet. If the channel is sensed free, the BluetoothLL sends the RTS packet to the Space. The piconet is sensed free by the node if it has received an ACK packet acknowledging the last data segment of an IP packet and no packets afterwards.

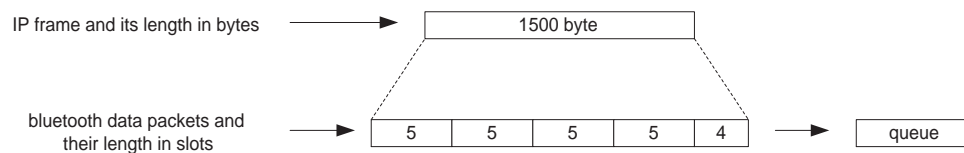


Figure 3.9: Segmenting an IP packet to Bluetooth data packets

Sending a Bluetooth packet means that the BluetoothLL calculates the time when the next slot begins and when it has to send the packet to the Space. After determined the

exact time of sending the BluetoothLL calculates the frequency used in the corresponding slot, applying the Bluetooth frequency selection kernel (Appendix A.3).

Receiving a Bluetooth packet means that the node checks whether the packet fits to its receive window and the packet frequency corresponds to the frequency which the node was listen to in the last slot. To do this, the BluetoothLL generates the frequency used in the current slot using the Bluetooth frequency selection kernel. The frequency check passes, if the calculated frequency is the same as the frequency of the packet. The packet fits the node's receive window, if the transmission started after the node switched on its receiver and before the node turned it off (Figure 3.10).

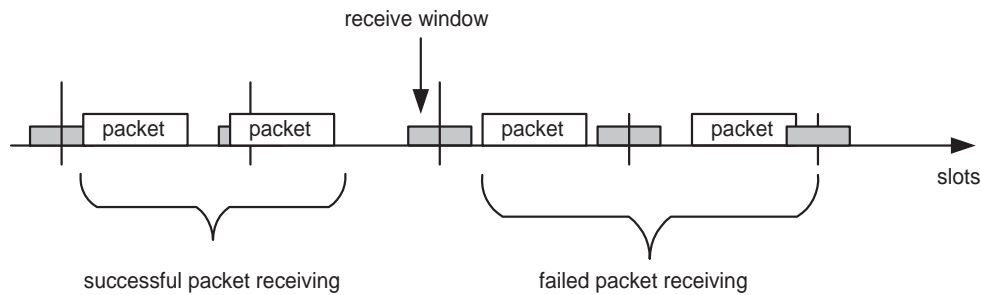


Figure 3.10: Receive window checking

After sending the RTS, the BluetoothLL waits for a CTS response in the next slot. If it receives a valid CTS packet, it starts to transmit the packets of the given IP packet from the queue. If there is no valid CTS response, the BluetoothLL retransmits the RTS using backoff (Section 3.11).

If the node receives an RTS packet, the BluetoothLL checks whether the node can accept the request to send. If the node has to do something else - for example inquiring, paging, changing to another piconet - the BluetoothLL will not send a CTS packet to the source. If the node can accept the request, the BluetoothLL sends a CTS response.

After a successful RTS-CTS exchange the source starts to transmit the packets in the queue. After every packet the source waits for an ACK packet from the destination, which acknowledges the correct reception. Only positive ACK-s are sent. If there is an error

during the reception of the data packet, the receiver responds with nothing. If no ACK is received, the source retransmits the data packet. If the source receives the ACK, it deletes the packet corresponding to the ACK and continues transmitting the queue with the next data packet. After the last ACK is received, the channel becomes free for other transactions (Figure 3.11).

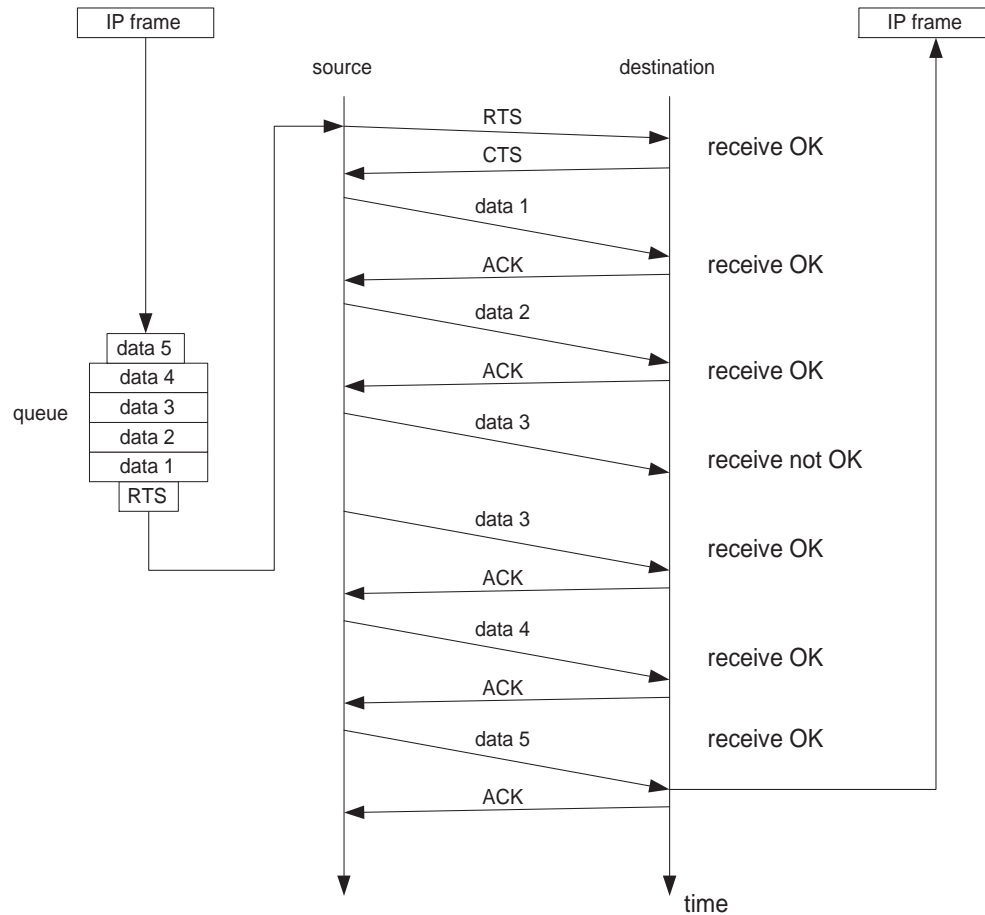


Figure 3.11: Transmitting the IP packet over the Bluetooth physical channel

In case of multi-slot packets, the node receives a packet with type header before receiving the whole data packet. This header packet tells the node to hold its receiver on the currently used frequency. If the node receives the whole data packet, it will not use the Bluetooth frequency selection kernel to calculate frequency, it checks instead, whether the frequency of the incoming packet is the same as it was told in the header. If a node receives the last data packet of an IP packet, it sends the IP packet to the upper layers.

Each node could have a different slot length parameter. With this parameter we take the clock drift into account. If a node calculates the time of packet sending, it calculates the length of the slots with the slot length parameter.

### 3.14 Upper Layers

To generate IP packets for the measurements I created a constant bit-rate (CBR) traffic generator. The input parameters of the CBR generator are the size and the numbers of the IP packets. We can see the network architecture with a CBR generator on Figure 3.12.

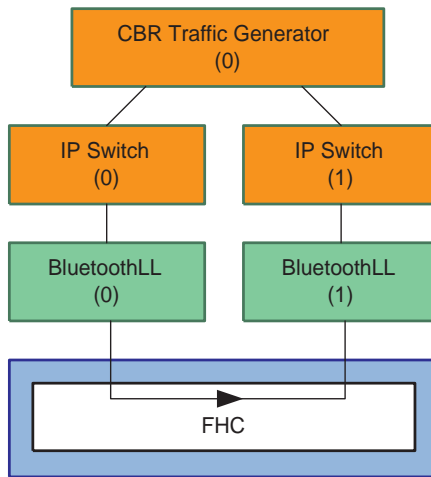


Figure 3.12: Network architecture with a constant bit-rate traffic generator

After starting the simulation the CBR generator creates an IP packet, and sends it to the BluetoothLL of the source node and waits until it receives the packet from the destination node. If the IP packet is received it generates and sends the next IP packet to the source node. This cycle is repeated until all IP packets are sent through the network.

## Chapter 4

# Simulation Based Analysis

In this chapter I analyze, using the developed simulation tool, the decrease in efficiency of the communication when interference is present. I investigate the case of one and four pair of nodes. I also analyze how the throughput depends on the IP packet size in case of 10% packet loss.

I introduce how to model the accuracy of synchronization and I make a numerical analysis on the accuracy of synchronization and power consumption. I show how we can decrease the power consumption of Bluetooth devices.

I also investigate the overhead of the protocol. I show how much time is spent with sending the RTS-CTS packets in case of several IP packet sizes and I investigate how much time is spent with backoff.

Finally I show the differences between several  $CV_{min}$  and  $CV_{max}$  settings.

### 4.1 Parameters and Network Structures

In the simulations I used one master and several slave nodes. Every node was a participant of the master's FHC. Since I investigated the traffic on the physical channel, I did not

used stand alone nodes during my simulations.

I investigated two cases of the network structure. In the first case I made the simulations with one pair of nodes (Figure 4.1), in the second case I created four pair of nodes (Figure 4.2). Each pair had its own traffic generator. In both cases the nodes were fixed, they did not move.

For the transmission rate I used 1 Mbit/s. The slot length parameter in every node was the same,  $625 \mu\text{s}$ .

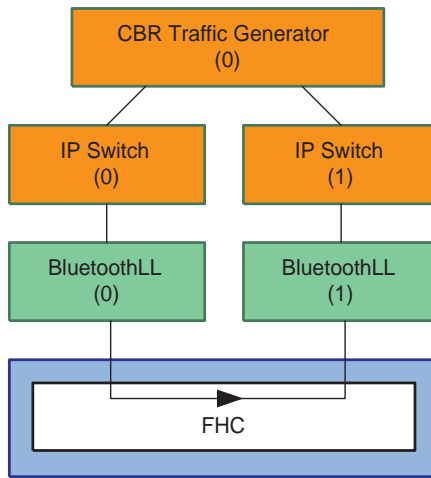


Figure 4.1: Network structure with one pair of nodes

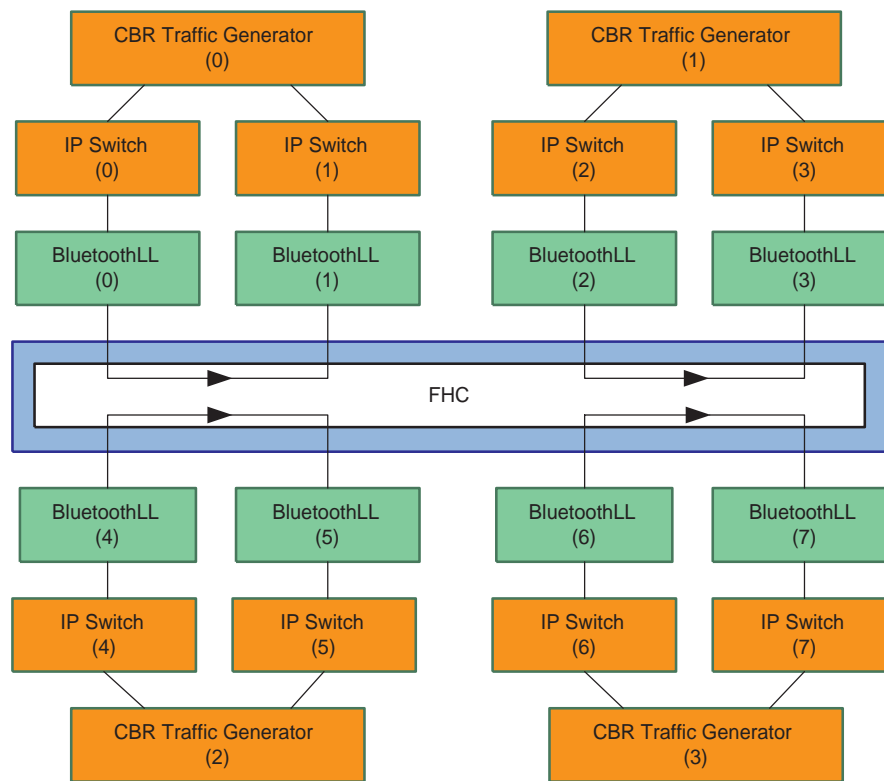


Figure 4.2: Network structure with four pair of nodes

## 4.2 Effect of External Interference

I investigated how the external interference influences the throughput in case of one and four pair of nodes. I used 40, 576 and 1500 bytes long IP packets in my investigations. These packet sizes are the typical IP packet sizes in today's networks [6].

In Figures 4.3, 4.4 and 4.5 we can see how the external interference influences the throughput. It is obvious that the higher the external interference is, the lower throughput values we get. If the packet loss is between 0 and 20% we get good throughput values compared

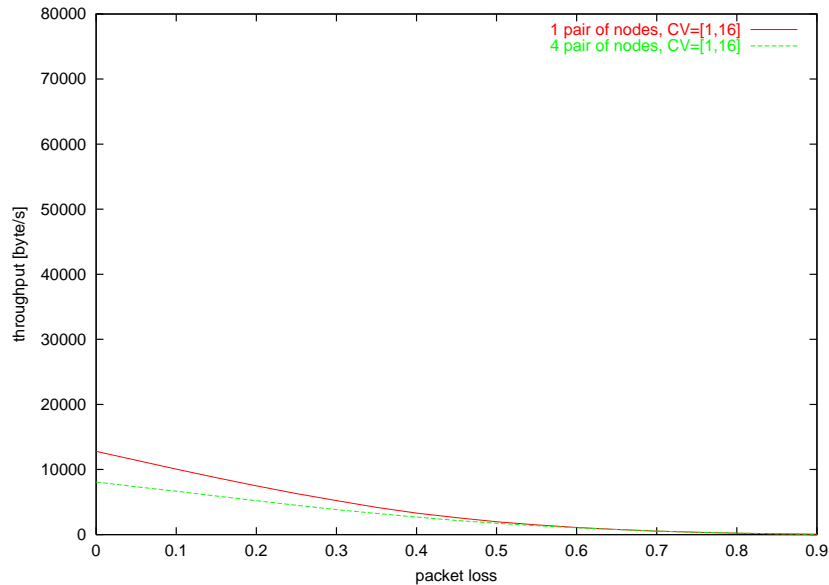


Figure 4.3: Throughput in case of one and four pair of nodes transmitting 50000 IP packets with the length of 40 bytes

to the higher packet loss ratios. The lines of the one and four pair of nodes case are close to each other, so there are not any significant differences in the accumulated throughput between the two cases.

On the figures we can see that in case of small IP packets (Figure 4.3) the throughput is very small compared to the case of large IP packets (Figures 4.4, 4.5). We can see that the size of IP packets has a significant influence to the throughput. Therefore I investigated in Section 4.5 how the IP packet size influences the throughput.



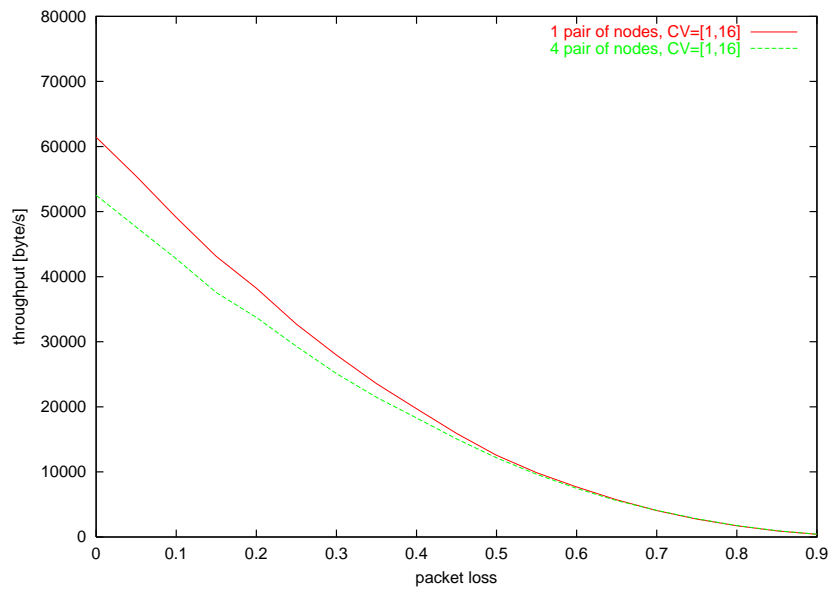


Figure 4.4: Throughput in case of one and four pair of nodes transmitting 3473 IP packets with the length of 576 bytes

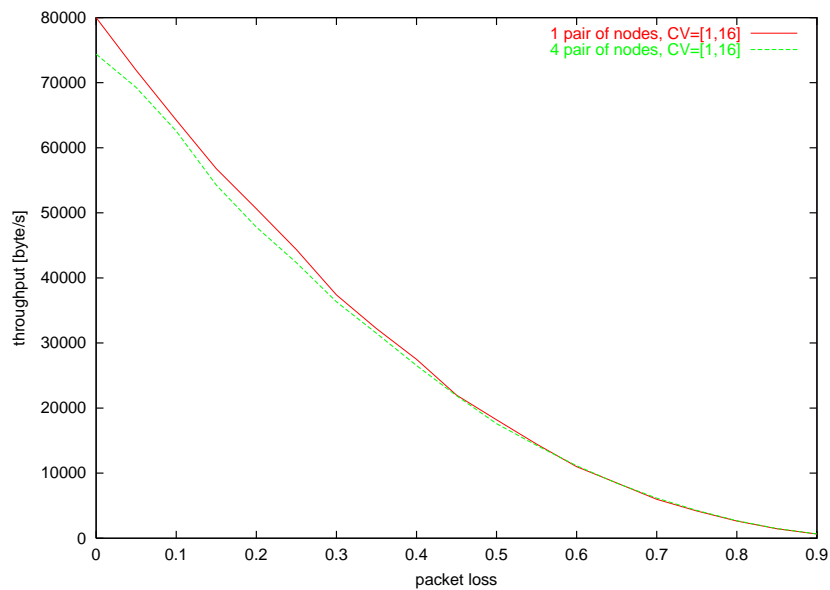


Figure 4.5: Throughput in case of one and four pair of nodes transmitting 1334 IP packets with the length of 1500 bytes

### 4.3 Modeling the Accuracy of Synchronization

The accuracy of synchronization can be modeled as follows. If we assume that a master sends a packet to a slave, the slave has to activate its receiver before and it should not turn its receiver off until the master starts sending the packet. In the equation  $n$  corresponds to the  $n$ th slot,  $sl$  corresponds to the slot length parameter,  $rw$  corresponds to the receive window.

$$n * sl_{slave} - \frac{rw}{2} < n * sl_{master} < n * sl_{slave} + \frac{rw}{2}$$

After arranging we get the following equation.

$$-\frac{rw}{2} < n * (sl_{master} - sl_{slave}) < \frac{rw}{2}$$

We can see that if the slot length in the master and in the slave are the same which means that their clocks are well synchronized, the receive window can be arbitrarily small. If the receive window is small, the power consumption is less, since the receiver is turned on only for a short time.

Although the receive window can be very small, it can not be zero. This is because there is propagation delay during packet transmission. If we assume that the range between the two nodes is 100 m (this is not typical but possible in Bluetooth), and the speed of the radio wave is  $3 * 10^8$  m/s the propagation delay can be calculated as follows

$$propagation\ delay = \frac{100\ m}{3 * 10^8\ m/s} = 0.333 * 10^{-6}\ s.$$

This means that if the value of the radio range between nodes is not higher than 100 m the  $\frac{rw}{2}$  has to be minimum  $0.333\ \mu s$ . In the real life the clocks are not well synchronized, so some clock drift should be allowed. To allow some clock drift the  $\frac{rw}{2}$  value is set to  $10\ \mu s$  in the Bluetooth specification.

## 4.4 Synchronization and Power Consumption

I investigated the power consumption as follows. Let us assume that one unit of power is consumed during one second while the receiver or the transmitter of the Bluetooth node is turned on. So if we measure the time spent with turning the transceiver on we measure the power consumption in unit/second-s. Let us investigate the case where we know the traffic pattern of the devices.

Regarded to the typical value the mutual clock drift of Bluetooth devices could be up to 40 ppm (parts per million). It means that the master has to send packets to the slaves at least every 0.25 seconds in order to keep synchronized them to the piconet, since the size of the slave's receive window in active mode is  $\pm 10 \mu s$ . If we assume that the master sends packet to the slaves in the piconet more frequently than 0.25 seconds, the receive window can be smaller than  $\pm 10 \mu s$ .

Let us investigate a piconet where four slaves are communicating with one master. The master sends five slot packets to the slaves periodically, and the slaves respond also with five slot packets. The sending of the five slot packet takes 8 slots, since before sending the data packet an RTS should be also sent, and receiving the CTS and ACK packets also takes 2 slots. So the communication between the master and a slave takes 16 slots. This means that one slave receives packet from the master with a period of 56 slot. The 56 slots take  $625 \mu s * 56 = 0.035$  seconds. During this time due to the clock drift the clocks can lose synchronization of  $0.035 s * 40 ppm = 1.4 \mu s$ . This means that in this case the receive window could be set to  $\pm 2 \mu s$ , which is 20% of the the default  $10 \mu s$ . Reducing the size of the receive window saves battery power.

We can use the model to determine the size of the receive window from the intensity of incoming packets.  $n$  corresponds to the number of slots where the slave receives packets from the master periodically. The  $625 \mu s$  is the length of one slot and the  $40 ppm$  is the mutual clock drift of the devices. The propagation delay is  $0.333 * 10^{-6} s$ .

$$n * 625 \mu s * 40 ppm + propagation\ delay < \frac{rw}{2}$$

In Figure 4.6 we can see the recommended size of the receive window in function of the intensity the slave receives packets from the master.

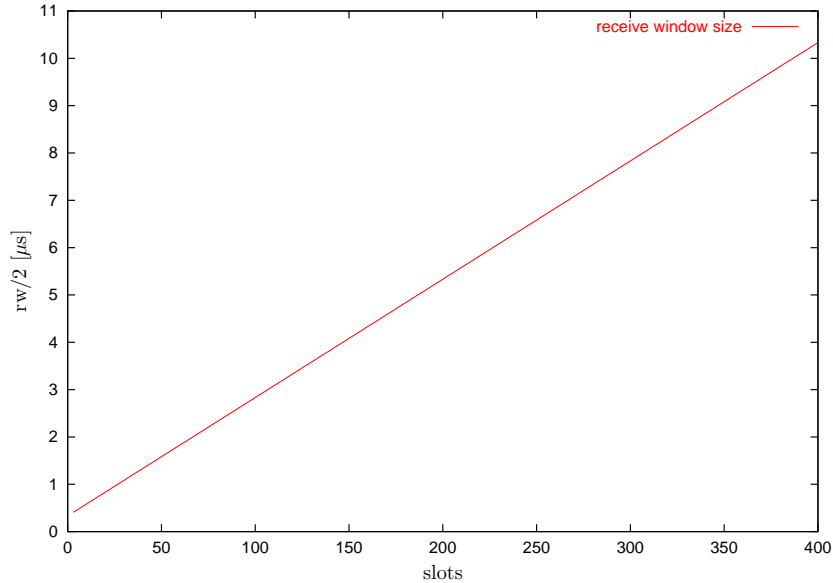


Figure 4.6: Receive window size in function of the intensity of incoming packets

We can say that the more often a slave receives packets from the master the better the synchronization can be and so the smaller receive window is needed. We can also say that the better the accuracy of the clocks is the smaller receive window can be used. And the smaller the receive window is the less power is consumed. It could be investigated in a future work, how we can dynamically set the receive window size to reduce the power consumption of Bluetooth devices.

#### 4.4.1 Power Consumption in case of Continuous Traffic

I made simulations to investigate the time spent with receiving and sending. In the simulations 1334 IP packets were transmitted with length of 1500 bytes. First I investigated the case of continuous traffic (Figure 4.7). This means that an IP packet is generated and sent to the channel immediately after the previous IP packet is received.

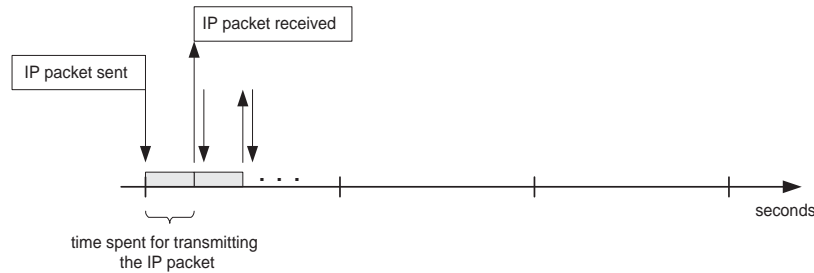


Figure 4.7: Continuous traffic

In Figures 4.8, 4.9 and 4.10 we can see how the time spent with receiving and sending depends on the receive window size in case of zero, 10% and 20% packet loss. We can see that in this receive window interval the time spent with receiving and sending is almost constant compared to the full time of the IP packet transmissions. In the case of zero packet loss (Figure 4.8), we can see that the time spent with sending by the source (node0) is almost the same as the time spent with receiving by the destination (node1). If the receive window were zero, these two lines were the same. We can see that the increase of the time spent with receiving due to the larger receive window is not significant in this interval.

In Figures 4.9 and 4.10 the packet loss is not zero, so packets can be lost due to external interference. We can see that the time spent with receiving by the destination is smaller than the time spent with sending by the source. This is because the source sends multi-slot packets to the destination, and this packets can be lost due to external interference. The destination spends time only with receiving the header of the packet, but the source spends time with sending the whole packet (see multi-slot packet sending in Figure 3.6).

In Figure 4.11 we can see how the time spent with receiving and sending depends on the

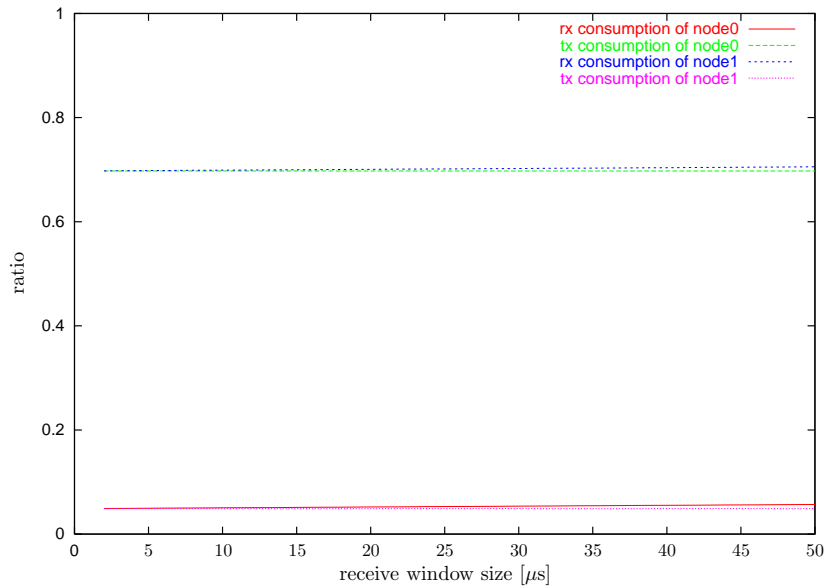


Figure 4.8: Time spent with receiving and sending compared to the full time of packet transmissions, one pair of nodes, 1334 IP packets with length of 1500 bytes,  $CV_{min}=1$ ,  $CV_{max}=16$ , 0% packet loss, continuous traffic

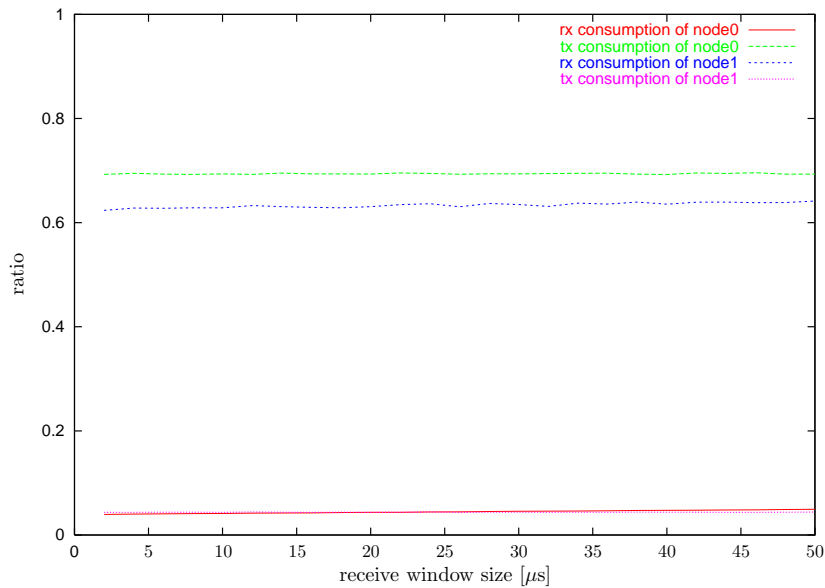


Figure 4.9: Time spent with receiving and sending compared to the full time of packet transmissions, one pair of nodes, 1334 IP packets with length of 1500 bytes,  $CV_{min}=1$ ,  $CV_{max}=16$ , 10% packet loss, continuous traffic

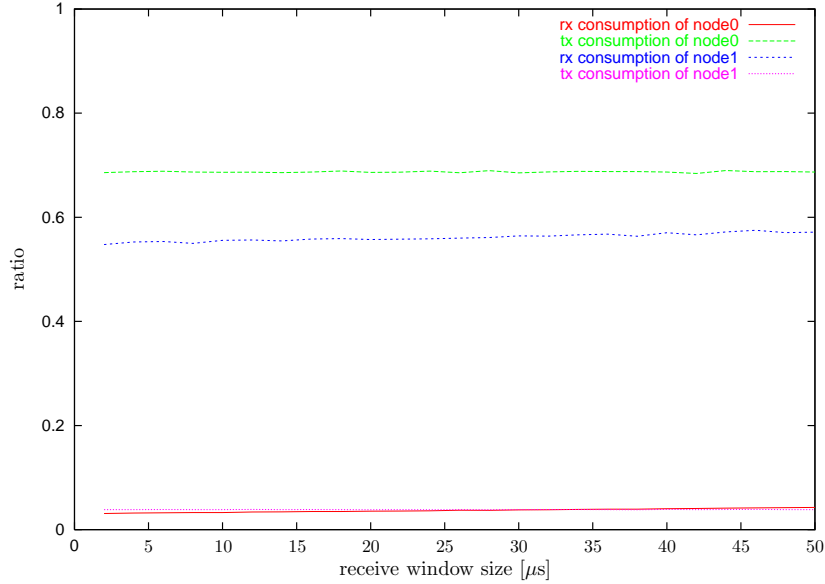


Figure 4.10: Time spent with receiving and sending compared to the full time of packet transmissions, one pair of nodes, 1334 IP packets with length of 1500 bytes,  $CV_{min}=1$ ,  $CV_{max}=16$ , 20% packet loss, continuous traffic

external interference. We can see that the time spent with receiving and sending decreases compared to the full time of transmissions. The decrease of the time of sending is because the sender spends time with backoff. During backoff the transmitter is turned off, but the full time of transmissions increases.

We can see that the time of receiving decreases in function of the external interference. The reason for this decrease is that while the source spends time with transmitting a lost packet, the destination turns its receiver off if it receives a corrupted packet header. So the source spends much more time with transmitting the packets than the destination with receiving.

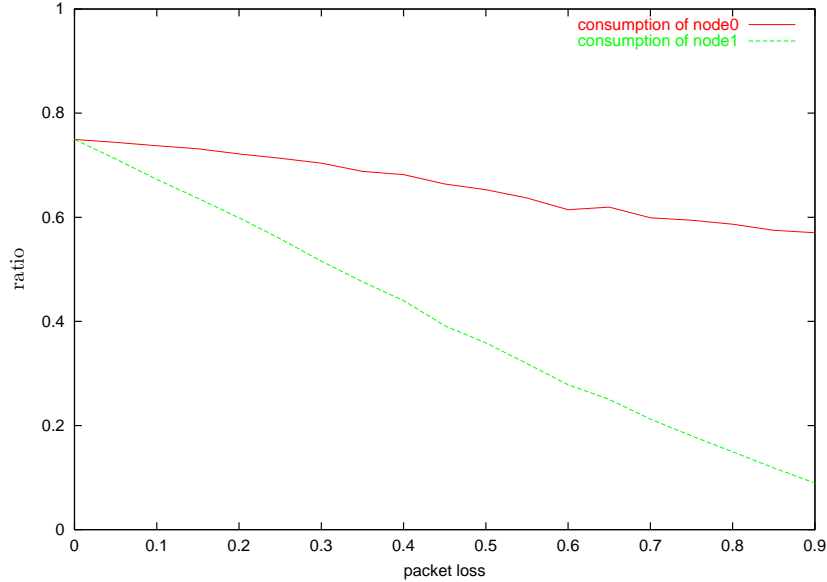


Figure 4.11: Time spent with receiving and sending compared to the full time of packet transmissions, one pair of nodes, 1334 IP packets with length of 1500 bytes,  $CV_{min}=1$ ,  $CV_{max}=16$ , 10  $\mu s$  receive window length, continuous traffic

#### 4.4.2 Power Consumption in case of Non-Continuous Traffic

I investigated the case of non-continuous traffic, as well. In this case IP packets are sent at every second (Figure 4.12) and not when the previous IP packet arrives. In the simulations park, hold and sniff modes were not used.

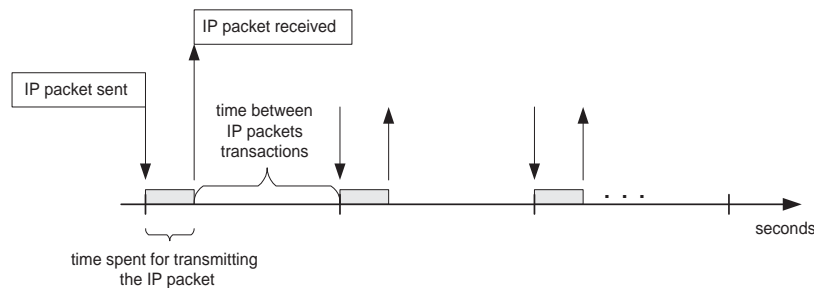


Figure 4.12: Non-continuous traffic

In Figures 4.13, 4.14 and 4.15 we can see how the time spent with receiving and sending depends on the receive window size in case of zero, 10% and 20% packet loss. The time spent with receiving and sending is compared to the time while IP packets are transmitted,



excluding the time between IP packets transactions. During the time between IP packets transactions the receiver still listens to incoming packets. We can see that the time spent with receiving increases in function of the receive window compared to the full time of transmissions.

We can see that the time spent with receiving and sending highly depends on the receive window size. The longer the receive window is the more power is consumed. We can see that it is recommended to use as short receive window as possible.

In the figures we can see that the higher the packet loss is, the smaller the time spent with receiving is compared to the time of transmitting the IP packets. This is because the same reason as described in the previous subsection. While the source spends time with transmitting a packet which is lost due to external interference, the destination receives the corrupted packet header and turns its receiver off. So the source spends much more time with transmitting the packets than the destination with the receiving.

We can see that the time spent with sending the IP packets is constant compared to the full transmission time. This is because the time spent with packet sending determines mainly the full transmission time.

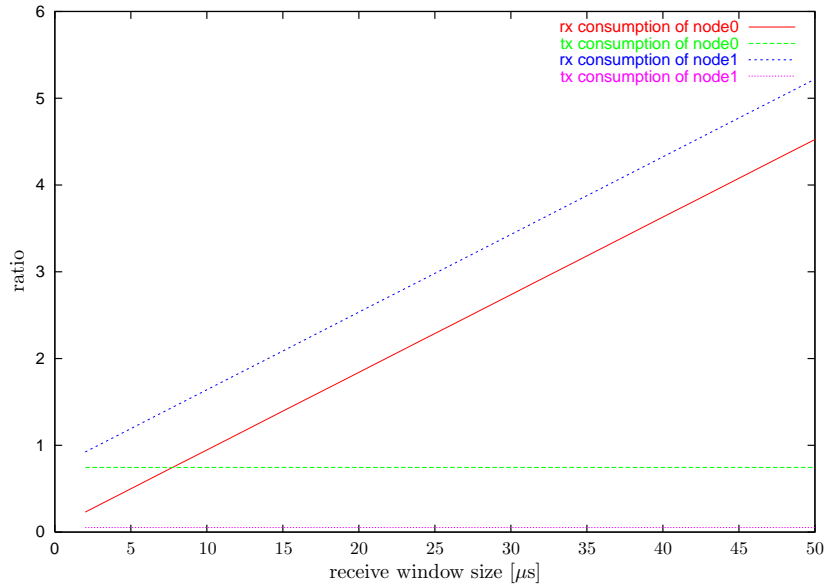


Figure 4.13: Time spent with receiving and sending compared to the time of transmitting the IP packets, one pair of nodes, 1334 IP packets with length of 1500 bytes,  $CV_{min}=1$ ,  $CV_{max}=16$ , 0% packet loss, non-continuous traffic

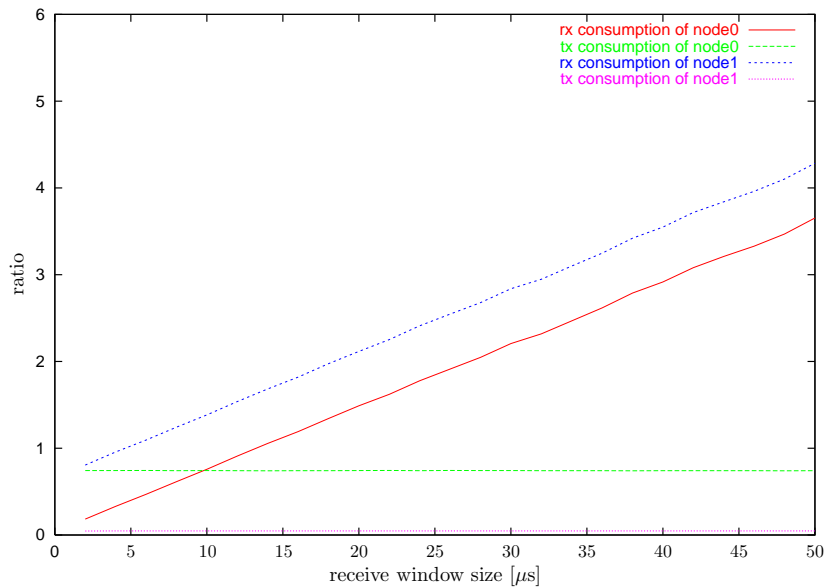


Figure 4.14: Time spent with receiving and sending compared to the time of transmitting the IP packets, one pair of nodes, 1334 IP packets with length of 1500 bytes,  $CV_{min}=1$ ,  $CV_{max}=16$ , 10% packet loss, non-continuous traffic

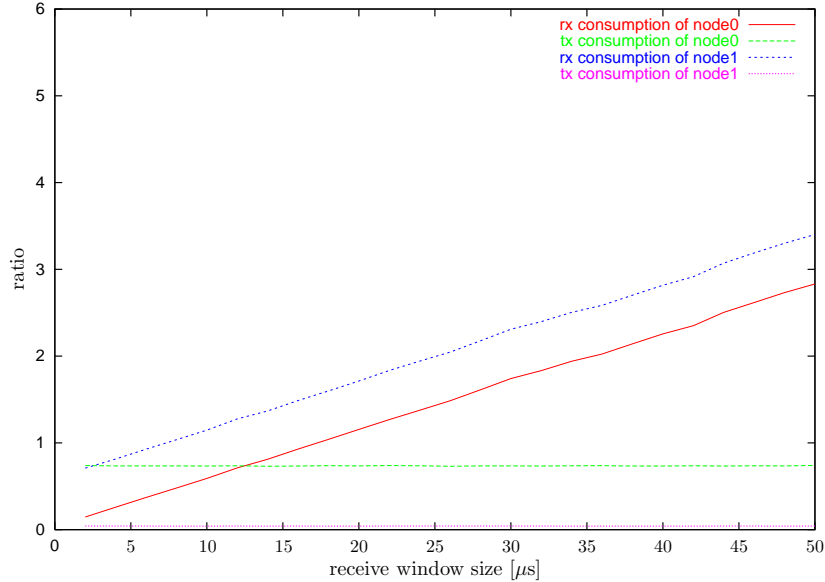


Figure 4.15: Time spent with receiving and sending compared to the time of transmitting the IP packets, one pair of nodes, 1334 IP packets with length of 1500 bytes,  $CV_{min}=1$ ,  $CV_{max}=16$ , 20% packet loss, non-continuous traffic

## 4.5 Effect of the IP Packet Size

In Figure 4.16 we can see how the IP packet size influences the throughput in case of 10% packet loss caused by external interference. Using small IP packets with size of 40 bytes the throughput is very low, it is about 25% of the maximum throughput of the protocol. This is because the overhead of the protocol is very high compared to the cases of 576 and 1500 byte long IP packets. In this case more RTS-CTS packets have to be exchanged to transmit the same amount of data. In case of 576 and 1500 byte long IP packets the throughput is about 79% and 85% of the maximum throughput. In Table 4.1 we can see the number of physical layer packets and slots which are required to transmit the IP packets if we want to transmit 2000000 bytes from one node to another. We can see that transmitting the same amount of data, in the case of 500 byte long IP packets the transmission uses 17.6% of the slots needed in the case of 40 byte long IP packets.

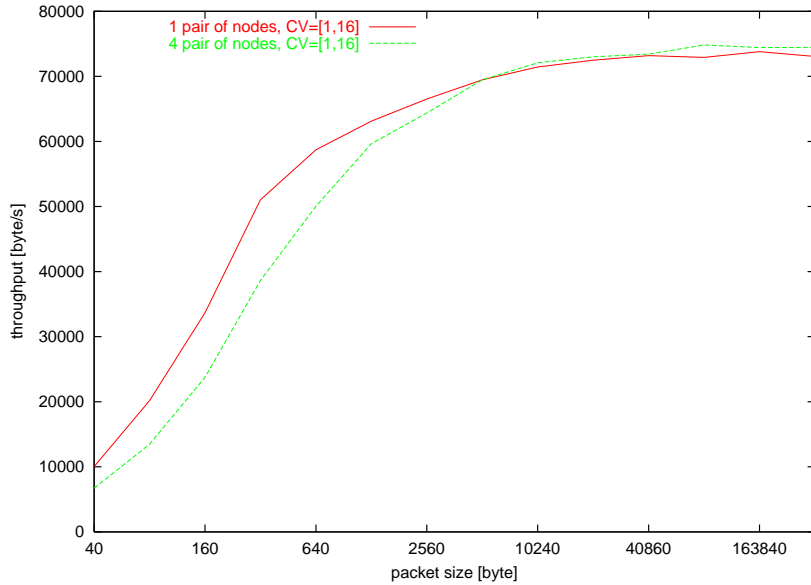


Figure 4.16: Throughput in function of packet size, one and four pair of nodes, 10% packet loss,  $CV_{min}=1$ ,  $CV_{max}=16$ , packet size in bytes

IP packet size	IP packets	RTS slots	CTS slots	Data slots	ACK slots	sum slots
40 byte	50000	50000	50000	$2*50000$	50000	250000
500 byte	4000	4000	4000	$8*4000$	4000	44000

Table 4.1: Slots needed to transmit 2000000 bytes using 40 and 500 byte long IP packets in the case of zero packet loss

## 4.6 Overhead of the Protocol

In Figures 4.17, 4.18 and 4.19 we can see the time spent with RTS-CTS sending and backoffing compared to the full time spent with transmitting the IP packets in case of 40, 576 and 1500 bytes long IP packets.

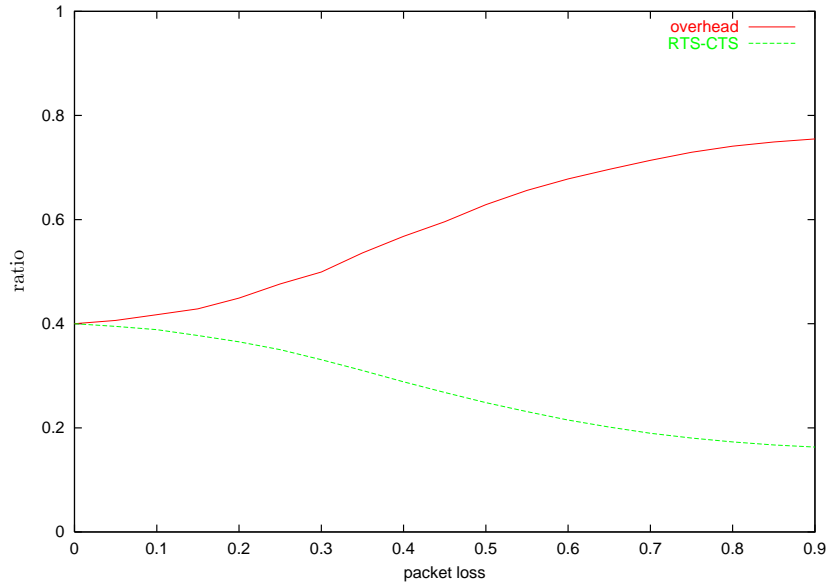


Figure 4.17: Protocol overhead transmitting 50000 IP packets with 40 bytes of length, one pair of nodes,  $CV_{min}=1$ ,  $CV_{max}=16$

As we can see in Figures 4.17, 4.18 and 4.19 the overhead is very high, more than 40% in case of small, 40 byte long IP packets. We can also see that using large, 576 and 1500 byte IP packets the overhead is much smaller than in case of small packets.

The time spent with backoff is the difference between the two lines in Figures 4.17, 4.18 and 4.19. We can say that in case of reasonable external interference (10%, 20% packet loss), the time spent with backoff is not significant. The backoff mechanism which is used in the simulator works well in case of every investigated packet sizes.

In Figures 4.20, 4.21 and 4.22 we can see the overhead of the protocol in case of four pair of nodes compared to the case of one pair of nodes. We can see, that the protocol overhead is much greater in case of four pair of nodes than in case of one pair of nodes.

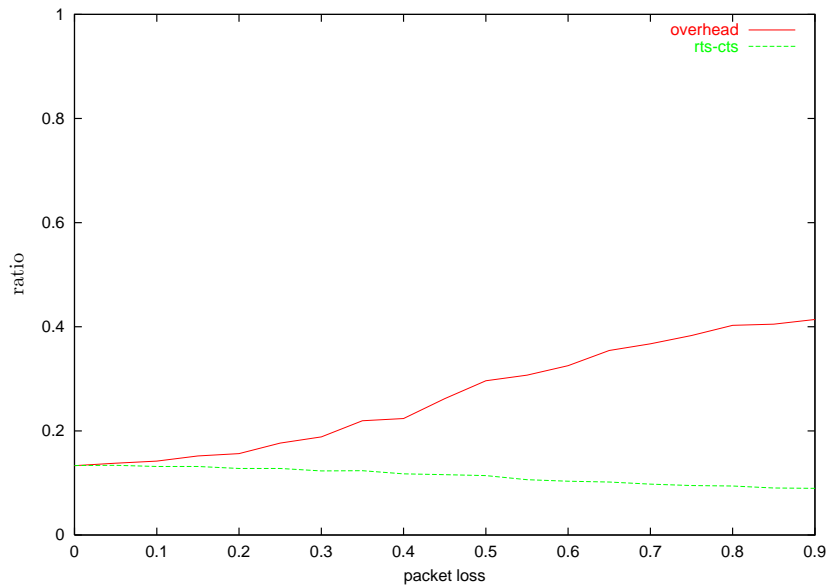


Figure 4.18: Protocol overhead transmitting 3473 IP packets with 576 bytes of length, one pair of nodes,  $CV_{min}=1$ ,  $CV_{max}=16$

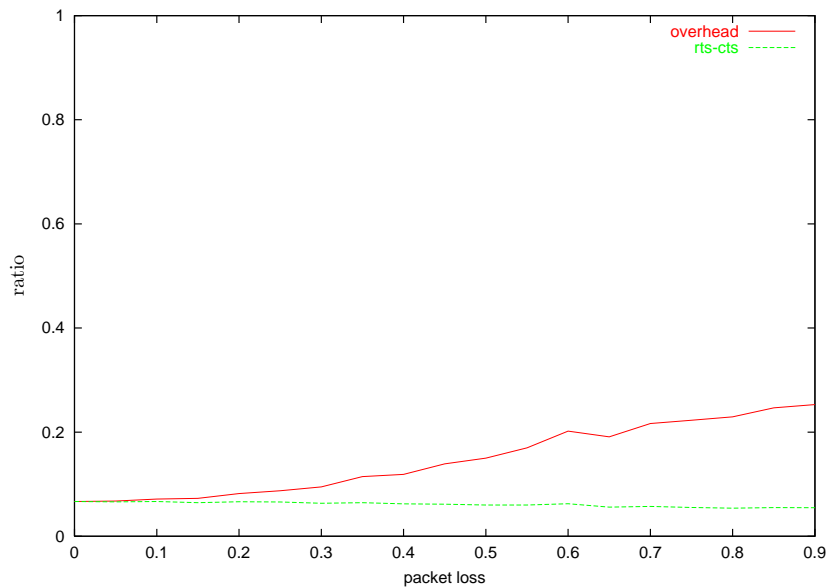


Figure 4.19: Protocol overhead transmitting 1334 IP packets with 1500 bytes of length, one pair of nodes,  $CV_{min}=1$ ,  $CV_{max}=16$

This is because the nodes spend more time with backoff, since every node wants to use the piconet in the same time, and so there are many packet collisions in the piconet.

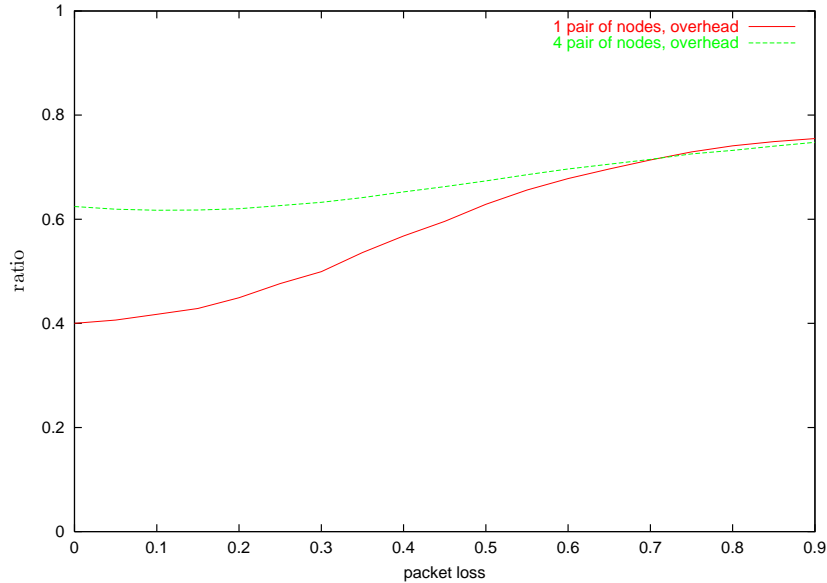


Figure 4.20: Overhead in case of four pair of nodes using 50000 IP packets with 40 bytes of length

In case of large IP packets (Figure 4.21 and 4.22) the overhead difference between the case of one pair of nodes and four pair of nodes are smaller than in case of small IP packets (Figure 4.22). This is because using large IP packets to transport the same amount of data generates less RTS packets in the piconet and so the possibility of packet collisions are less than in case of small IP packets where more RTS packets are generated.

In the figures we can see that in case of high packet loss ratios the two scenarios have the same overhead. This is because in case of high packet loss ratios so many packets are lost, that it does not count whether the packets are lost due to external interference or due to packet collisions.

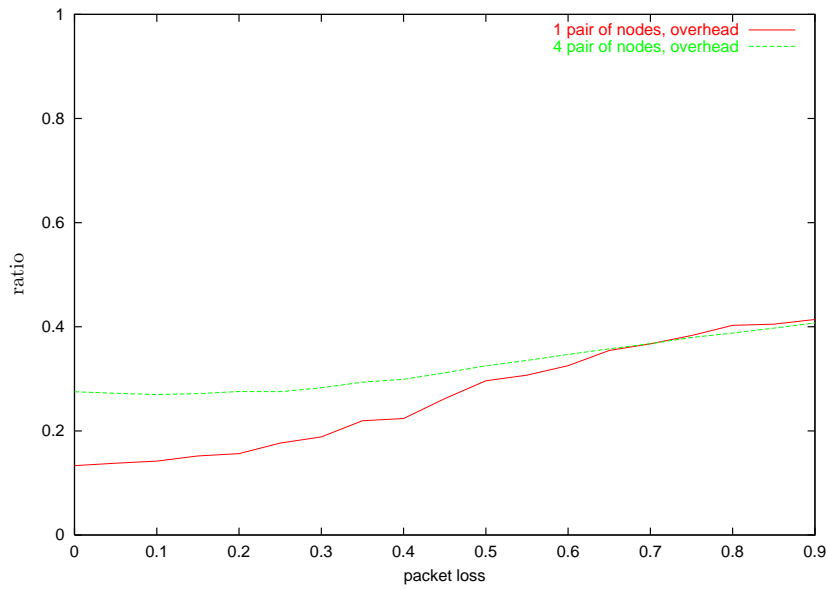


Figure 4.21: Overhead in case of four pair of nodes using 3473 IP packets with 576 bytes of length

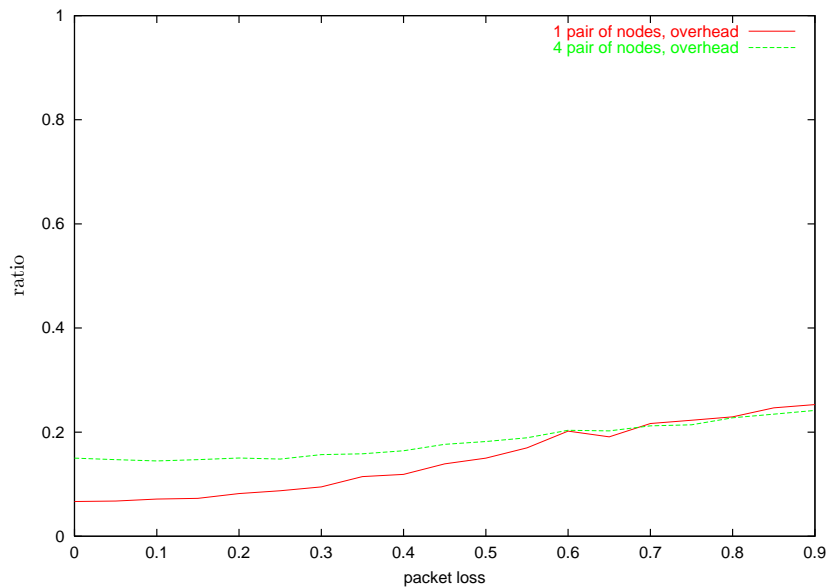


Figure 4.22: Overhead in case of four pair of nodes using IP packets with 1500 bytes of length



## 4.7 Several $CV_{min}$ and $CV_{max}$ values

In Figure 4.23 we can see the throughput as a function of the packet loss using different values for the  $CV_{min}$  and the  $CV_{max}$  constants. We can see that there could be significant differences in the throughput between the different settings of the  $CV_{min}$  and  $CV_{max}$  constants. It could be investigated in a future work, how the throughput and the overhead of the protocol depends on the different settings of these constants.

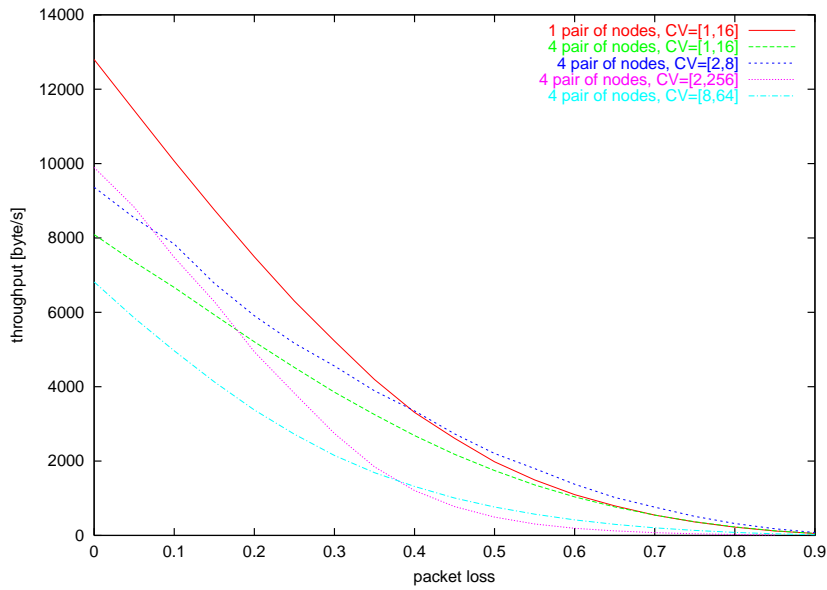


Figure 4.23: Throughput in function of the packet loss, different  $CV_{min}$  and  $CV_{max}$  values

## Chapter 5

# Conclusions

In this thesis I overviewed the Bluetooth technology. I have showed how Bluetooth provides protection against external interference, and how it maintains the synchronization between nodes.

I have developed a simulation tool for modeling the Bluetooth physical channel on packet-level. This tool is capable of modeling the external interference, the accuracy of synchronization and the power consumption of nodes. For the investigations I have created a simplified link layer which is based on a random access method.

I have shown how the external interference decreases the efficiency of the communication between nodes. I have made a numerical analysis on how the power consumption of Bluetooth nodes depends on the accuracy of synchronization and how it can be decreased by changing the receive window size in the knowledge of the traffic pattern. I show how the power consumption depends on the receive window size and on the external interference. In case of continuous traffic the power consumption is not influenced significantly by the size of the receive window. In case of non-continuous traffic the power consumption highly depends on the receive window size, the longer the receive window is the more power is consumed.

I have analyzed how the throughput depends on the IP packet size and on the external

---

interference. The longer the transmitted IP packets are the better throughput values we get. I have investigated how the overhead of the implemented protocol and the time spent with backoff depends on the external interference. In case of 0-20% packet loss the time spent with backoff is not significant. Finally I have showed how the  $CV_{min}$  and  $CV_{max}$  parameters of the backoff mechanism influences the throughput.

The results show that the Bluetooth technology is robust against external interference and can be efficient in power consumption. The results also show that the created simplified link layer, which is based on a random access method works well on the Bluetooth physical layer. In addition the simulator is capable of investigating new solutions for Bluetooth.

# Bibliography

- [1] Bluetooth Baseband Specification version 1.1, <http://www.bluetooth.com/>
- [2] Jaap Haartsen: *Bluetooth - The universal radio interface for ad hoc, wireless connectivity*, Ericsson Review No.3, 1998, pp. 110-117
- [3] György Miklós: *Bluetooth: Olcsó, drótnélküli helyi összeköttetés (Bluetooth: cheap wireless local connectivity)*, Magyar Távközlés, September 2000
- [4] Stefan Zürbes, Wolfgang Stahl, Kirsten Matheus, Jaap Hartsen: *Radio Network Performance of Bluetooth*, Proceedings of ICC 2000, New Orleans, 18-22 June, 2000.
- [5] dr. Imre Sándor: *Bevezetés a mobil számítástechnikába*, <http://www.hit.bme.hu/mcl/hu/oktatas.html>, 2000
- [6] Attila Vidács, József Barta, Zsolt Kenesi and Tamás Éltető: *Measurement-Based WWW User Traffic Model for Radio Access Networks*, Proceedings of 7th International Workshop on Mobile Multimedia Communications (MoMuC 2000), Tokyo, Japan, 23-26 October, 2000.
- [7] Patrick J. Megowan, David W. Suvak, Charles D. Knutson: *IrDA Infrared Communications: An Overview*, Counterpoint Systems Foundry Inc., <http://www.irda.org>
- [8] Mobile Ad-hoc Networks Working Group of IETF, <http://www.ietf.org/html.charters/manet-charter.html>
- [9] Internet Engineering Task Force, <http://www.ietf.org>

- 
- [10] András Rácz, György Miklós, Ferenc Kubinszky, András Valkó: *A Pseudo Random Coordinated Scheduling Algorithm for Bluetooth Scatternets*
- [11] György Miklós, András Rácz, Zoltán Turányi, András Valkó, Per Johansson: *Performance Aspects of Bluetooth Scatternet Formation*, MobiHOC poster session (First IEEE Annual Workshop on Mobile Ad Hoc Networking and Computing), August 2000
- [12] B. Raman, P. Bhagwat, S. Seshan: *Arguments for Cross-Layer Optimization in Bluetooth Scatternets*, Symposium on Applications and the Internet (SAINT'01), Jan 2001
- [13] T. Salonidis, P. Bhagwat, L. Tassiulas, R. LaMaire: *Proximity Awareness and Ad Hoc Network Establishment in Bluetooth*, Technical Report, February 2001,  
<http://www.isr.umd.edu/CSHCN/>
- [14] Zs. Haraszti, I. Dahlquist, A. Faragó, T. Henk: *PLASMA - An Integrated Tool for ATM Network Operation*, Proceedings of International Switching Symposium, 1995.

# Appendix A

## Simulator Specification

### A.1 Messages between Space and Bluetooth Nodes

**CreateFHCMsg** (nodeID, fhcID, xPos, yPos, pLoss, slotLength); Creates one FHC within Space. Packets in a FHC may be lost according to some simple loss model. The simplest is an iid loss with a constant probability Ploss. This can be used to model external interference. After creating the FHC the Space returns with fhcID.

**JoinFHCMsg** (masterID, nodeID, fhcID, xPos, yPos); The node joins the FHC identified by MasterId. From this time on the node will get every messages sent on the piconet which are in the node's radio range. The node should make sure that it joins only one FHC at a time. After joining to the FHC controlled by masterID the Space returns with fhcID.

**LeaveFHCMsg** (fhcID, nodeID); From this time on the node will not get any messages from the left FHC. The node could join to another FHC or the SC.

**LocationUpdateMsg** (nodeID, xPos, yPos); If the node moves, it has to update its location information. Every message containing the information xPos and yPos updates the location. So no extra LocationUpdate has to be sent.

**GetNeighboursMsg** (nodeID, fhcID, neighbourIDList); With this message the node can ask the Space about its neighbours in radio range. It is needed until inquiry and page procedures are not implemented.

**PacketMsg** (packet); Send the packet to the Space or to the node. Information contained in a packet data structure:

- sourceNodeID: the RECORD\_ID of the source node
- destinationNodeID: the RECORD\_ID of the destination node
- sourceFHCID: the RECORD\_ID of the FHC
- type: packet type could be data, header, ACK, beacon, RTS, CTS
- header: packet header length in bits
- payload: user payload length in bits
- remainingBytes: remaining bytes from the IP frame
- ipPacketID: the packet ID of the corresponding IP packet
- slots: 1: one slot packet; 2, 3, 4, 5: multislot packet
- bits: packet length in bits
- txRate: how many microseconds are needed to transfer one bit
- length: length in time
- frequency: the used frequency transmitting the packet
- xPos, yPos: the position of the source node
- sendingTime: time of sending
- currentSlot: the number of the sending slot
- currentQID: the queue ID at the source

- `bbePacketID`: the packet ID in the BluetoothLL
- `nextPacketID`: the ID of the next packet in the queue
- `sumSlots`: used in RTS packets allocating slots for data transfer
- `ackOK`: OK:1, not OK:0, it can be only OK, since only positive ACKs are used in the simulator
- `headerSent`: in case of multislot packets the Space sends a header before sending the whole packet, if the header is already sent
- `channelState` (set by the Space): 0: NORMAL, 1: NOISE

The channel state is determined separately for each member of the FHC and may have the following values:

- **NORMAL**: The packet can be successful received.
- **NOISE**. There is something in the channel but it cannot be correctly decoded. Could be due to collision in the piconet, external interference, or because of fading or path loss. If there is a collision at the receiver, the Space sets the ChannelState to NOISE.



## A.2 Packet types

Slots	FEC	Header	Pay. h.	Max pay.	CRC	Sum	FEC	Sum	Safety
1	1	126	8	136	16	160	80	366	259
1	0	126	8	216	16	240	0	366	259
2	1	126	16	548	16	580	290	996	254
2	0	126	16	838	16	870	0	996	254
3	1	126	16	968	16	1000	500	1626	249
3	0	126	16	1464	16	1496	0	1622	253
4	1	126	16	1380	16	1412	710	2248	252
4	0	126	16	2090	16	2122	0	2248	252
5	1	126	16	1792	16	1824	915	2865	260
5	0	126	16	2712	16	2744	0	2870	255

Table A.1: Data packets used in the simulator

Slots	FEC	Header	Pay. h.	Max pay.	CRC	Sum	FEC	Sum	Safety
1	1	126	8	136	16	160	80	366	259
1	0	126	8	136	16	160	0	286	339

Table A.2: RTS, CTS packets used in the simulator

Slots	FEC	Header	Pay. h.	Max pay.	CRC	Sum	FEC	Sum	Safety
1	1	126	0	0	0	0	0	126	499
1	0	126	0	0	0	0	0	126	499

Table A.3: ACK packets used in the simulator

### A.3 Bluetooth Frequency Selection

I implemented the selection kernel of the 79-hop system (Figure A.1). I used the connection state values [1] for the input parameters (Table A.4).

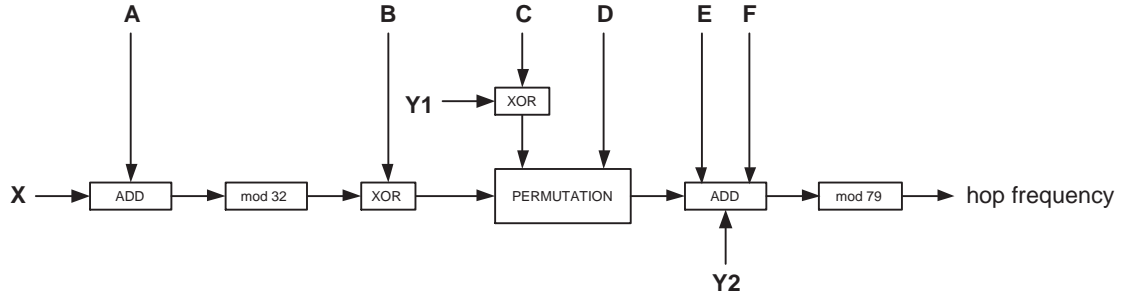


Figure A.1: Bluetooth frequency selection kernel for 79-hop systems

X	$CLK_{6-2}$
Y1	$CLK_1$
Y2	$32 \times CLK_1$
A	$A_{27-23} \oplus CLK_{25-21}$
B	$A_{22-19}$
C	$A_{8,6,4,2,0} \oplus CLK_{20-16}$
D	$A_{18-10} \oplus CLK_{15-7}$
E	$A_{13,11,9,7,5,3,1}$
F	$16 \times CLK_{27-7} \bmod 79$

Table A.4: Frequency selection kernel parameters