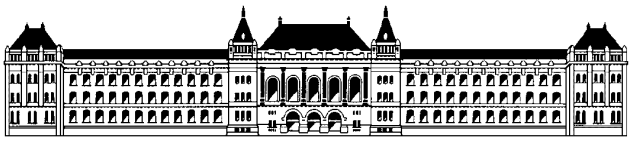

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications

Development and Evaluation of a New Bluetooth Neighbour Discovery Procedure

Eszter Kail

Master's Thesis

Advisors:

György Miklós
Miklós Aurél Rónai
M.Sc., Ericsson Research Hungary

Sándor Imre
Ph.D., Budapest University of Technology and Economics

Budapest, 2002.

Abstract

Bluetooth is a cheap and robust radio system that allows electronic devices to communicate within a short range in the ISM band. It was designed especially to eliminate cables between phones and headsets, and PCs and mouse and so forth.

In this thesis I investigate the connection establishment procedure of Bluetooth Technology. It is a two stage procedure, starting with the optional inquiry procedure which helps the master node to discover its neighbours, and continues with the page procedure which helps to set up an actual connection.

I also introduce a new flexible neighbour discovery (FND) procedure. It is based on beacon packets sent by the nodes. I study the performance of this latter algorithm. I compare the two procedures with analytical and simulation results and show that the FND procedure needs less time to discover a neighbour with a probability of 90%, therefore it is more suitable in systems where a long neighbour discovery or neighbour management procedure can not be afforded. I also carry out simulations in system where both symmetric and assymetric roles are assumed.

Finally I investigate the two procedures in a dynamically changing system and show that the FND procedure gives a more reliable solution in mobile systems.

Contents

Chapter 1

Introduction

There is an increasing need to eliminate cables between electronic devices. People use more and more portable devices to keep in touch with friends and business partners. The development of wireless technology is becoming faster and faster and it has shown us that it makes the use of devices very flexible, even though the usage of cables enables higher transmission rates.

There are different methods to replace cables between devices. Infrared links (IrDA) [3] are already used wide-spread all over the world to connect portable devices. Infrared links allow two devices to directly communicate within a few meters of each other. It is a very cheap solution, but infrared links have limited range and are sensitive to direction.

Another alternative to connect devices wirelessly is to use radio technology. Compared to infrared links radio technology has several benefits. Radios can work within a much greater range, and can connect more devices simultaneously.

Bluetooth technology [1] specifies such a universal short range radio interface working unlicensed in the 2.4 GHz ISM band. It was designed especially to eliminate

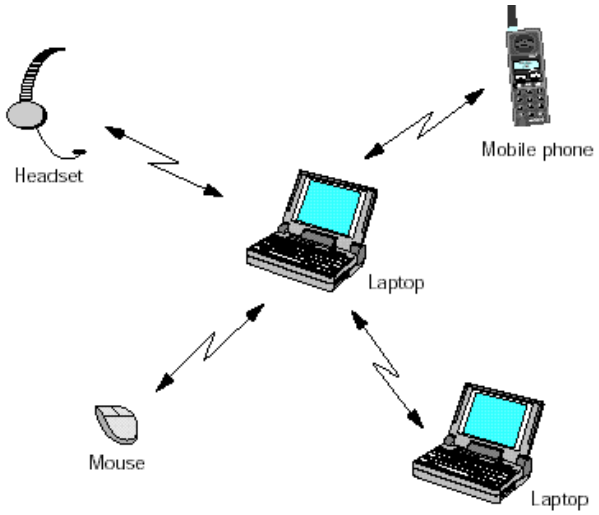

Figure 1.1: Cable replacement

cables between phones and headsets, PCs and mice and so forth (see Figure 1.1).

Bluetooth radios are cheap and small enough to be fitted inside any electronic device that may need to provide connectivity. It uses a frequency hopping spread spectrum (FHSS) method to avoid interference. The frequency hopping rate is 1600/sec. The advantage of this method lies in not only the resistance to interference (see Figure 1.2) but also in its simplicity and the possibiltiy to be implemented at low cost. With the help of this method it is also guaranteed that a great number of independent applications can operate in the same coverage area with limited interference.

Bluetooth uses a connection oriented approach. To start a connection there is a two stage procedure. The first stage, which enables the nodes to discover their neighours is optional and is called the inquiry procedure, while the other one, the paging procedure, is compulsory in order to synchronise the nodes with the frequency hopping method.

Bluetooth units have asymmetric roles. The node initiating the paging procedure becomes the master, controlling the traffic on the channels, while the others act as

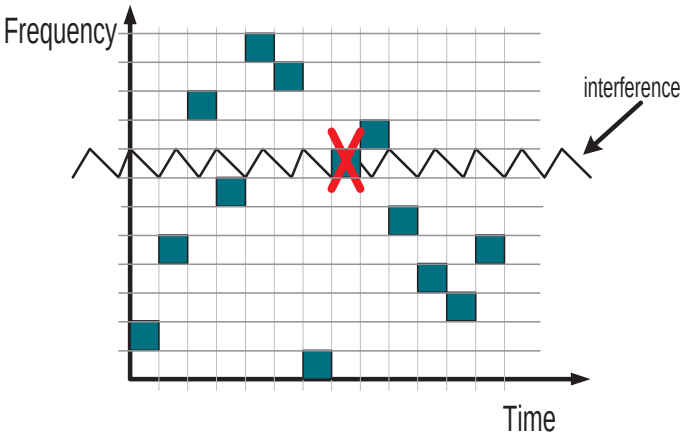

Figure 1.2: Frequency hopping method's resistance to interference

slaves. The slaves can only communicate with the master.

This solution is appropriate in office scenario, or where all the devices are different and play different roles. It is also true, that using the inquiry and paging procedure to update the information about the neighbours the setup of a connection would last relatively long meanwhile it would consume most of the capacity of the nodes participating in the procedures.

In this work I implement and describe the behaviour of a new way of discovering neighbours which was proposed in [9]. This new mechanism exploits all the advantages of the Bluetooth technology, yet enables peer nodes to exchange data packets and at the same time carry out this Flexible Neighbour Discovery (FND) in a time multiplex fashion. To compare the Inquiry and FND procedures I implement both of these neighbour discovery algorithms in the Plasma Bluetooth module [12]. I carry out a performance analysis of the FND procedure and I also give a brief comparison on how these two ways of neighbour discovery work in different circumstances in static and in dynamically changing system as well.

1.1 Related Work

As it was mentioned in the previous section wireless technology is becoming more and more important. Wireless LANs (Local Area Network) are the first approach to use wireless links in a local area. The major motivation and benefit from wireless LANs is the increased mobility. Compared to conventional network connections, network users can move almost without restriction within the coverage area. In addition to increased mobility, wireless LANs offer increased flexibility as well.

The IEEE 802.11 wireless LAN standard [10] places specifications on the parameters of both the physical and medium access control (MAC) layers of the network. The physical layer, which actually handles the transmission of data between nodes, can use either direct sequence (DS) spread spectrum, frequency-hopping (FH) spread spectrum, or infrared (IR) pulse position modulation. In the 802.11 standard there are two different modes to configure a network: ad-hoc mode and infrastructure mode. In the infrastructure mode the network is based on a cellular architecture where the system is subvided into cells. In each cell a specific device called Access Point (AP) controls the communication. To join an existing cell a device needs to get synchronisation information from the AP. Compared to the infrastructure mode in ad-hoc mode the network topology is not centralized. The node which wants to communicate with others scans for synchronisation information from the possible partners.

Synchronization can be done in two different ways, with passive and active scanning. In passive scanning the node waits until a special synchronization packet called beacon packet is received from the AP or in ad-hoc mode from an other device, while in active scanning the node tries to find an AP or other nodes by transmitting probe request packets and waits for the answer. After the connection is established the nodes need to keep synchronisation with the AP or each other to be able to follow the hopping sequence. It is also performed by beacon packets.

When a node wants to transmit a data packet, it first listens to be sure that no other node is transmitting in the same area. If the channel is clear the node transmits the packet. Otherwise, the node postpones the transmission to a later time. This mechanism is called as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol. Since a single channel is used throughout the network, nodes within an overlapping radio area can not communicate simultaneously.

As it is mentioned in the previous section Bluetooth [1] is an other wireless technology. It also uses the frequency hopping (FH) spread spectrum method with the extension that it allows to communicate more than one pair of nodes simultaneously within an overlapping radio area. Nodes in radio range can form an ad-hoc connection called piconet. One unit of the piconet becomes the master of the piconet while all others become slaves. The master node determines the frequency hopping pattern of the piconet. Bluetooth units which are slaves have to be synchronised to the master node. For the neighbour discovery Bluetooth describes an inquiry and a page procedure. The inquiry is an asymmetric and rather long procedure which can be carried out when the roles among the nodes are predefined. The nodes can form an ad–hoc network which is called scatternet. Several papers deal with topology construction and scatternet formation problem.

Theodoros Salonidis et al. in [7] give a new approach to link establishment in frequency hopping wireless systems. They suggest a new symmetric way to establish connections for Bluetooth units. It uses the Bluetooth inquiry procedure, but instead of predefining the roles, the nodes are performing the inquiry and scanning in an alternating fashion. It is a mechanism that guarantees an ad hoc point to point connection between two Bluetooth devices. It can be extended for more than two devices as well. They analytically calculate the mean and the variance of the link formation delay depending on the state residence time which follows a random distribution.

The Bluetooth Topology Construction Protocol (BTCP) is introduced in [7]. This protocol starts with nodes that have no knowledge about each other and terminates with the formation of a connected network. They showed that using this protocol the increase of the number of the nodes participating in the system does not affect the performance, so the nodes do not have to wait longer till the connection is established as the number of nodes increase.

A new scatternet formation algorithm is proposed in [8]. With the help of this protocol an ad-hoc network can be set up. This algorithm starts with isolated nodes and ends up with a complete network, with minimum number of piconets in it. The algorithm is based on the idea that every node performs the inquiry and the scanning with a certain probability.

In [6] András Rácz et al. propose the Pseudo-Random Coordinated Scatternet Scheduling (PCSS) algorithm to perform the scheduling of both intra– and inter–piconet communication. It introduces checkpoints which are meeting points when a master and a slave can communicate. If there are any packets to send they can exchange packets at this checkpoint. The checkpoints are selected by a pseudo random generator and are derived from the clock of the master and the MAC address of the slave.

A new way to exploit many of the advantages of the frequency hopping spread spectrum method is proposed in [9]. This protocol is called Multiple Frequency Hopping Channels (MFHC). It also uses the CSMA/CA random access scheme for each channel. The MFHC uses a similar way of neighbour discovery like the 802.11. It is based on beacon packets and on passive scanning.

1.2 Structure of the Thesis

In my thesis I first describe the main features of the Bluetooth technology, and in a detailed manner the Inquiry procedure.

In Chapter 3 I present the Flexible Neighbour Discovery (FND) procedure.

In Chapter 4 I investigate the performance of the Flexible Neighbour Discovery (FND) procedure. This chapter is started with the introduction of the simulation environment and the model I implemented. It is followed by simulation and analytical results, and finally I analyze the performance of this procedure in a dynamically changing environment.

In Chapter 5 I compare the Inquiry and the Flexible Neighbour Discovery procedure from two different points of view. First I investigate the overall capacity consumption of the nodes participating in a system, where there are asymmetric roles. The comparison in the second case was carried out in a system, where peer nodes were communicating.

In Chapter 6 I conclude my work.

Chapter 2

Bluetooth Technology

As mentioned in the introduction Bluetooth is a cheap, robust and power efficient radio technology that allows electronic devices to communicate wirelessly within a short range [1] [2] [4].

2.1 Piconet Concept

Two or more units that are within range of each other and share the same physical channel form a piconet. The unit controlling the traffic on the channel, acts as the master of the piconet, whereas the others act as slaves (see Figure 2.1). There can be up to seven active slave nodes in one piconet.

2.2 Physical Channel

Bluetooth operates worldwide at 2.45 GHz, which is the Industrial Scientific Medical (ISM) band. Since this band is license-free and globally available, radio systems

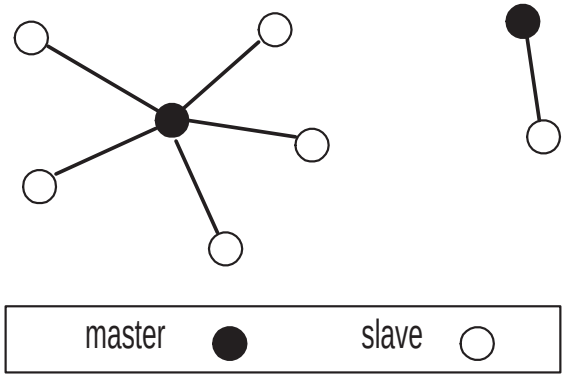

Figure 2.1: Piconets

using this frequency have to cope with interference caused by other devices. The ISM band ranges from 2,400 to 2,483.5 MHz in the USA, and in most parts of Europe but it is much narrower in Japan, Spain and France. Consequently, in countries where this band is wide 79 RF channels were defined, spaced 1MHz apart, while in other places there could be only 23 RF channels defined (see Table 2.1).

Modulation	G-FSK, h≤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	≤ 20 dBm

Table 2.1: Radio parameters

To avoid interference, Bluetooth uses the Frequency Hopping Spread Spectrum (FHSS) method with a hopping rate of 1600 hops/s. It means that the channel is divided into time slots, each 625 μs long and in every slot a different frequency is used.

All Bluetooth units participating in one piconet are time- and hop-synchronized to the channel. That means every unit in the piconet uses the master address and clock to follow the hopping sequence and calculate the appropriate frequency for the next

slot. Since each slave in the piconet has its own native clock, it has to add an offset to its clock shown in Figure 2.2.

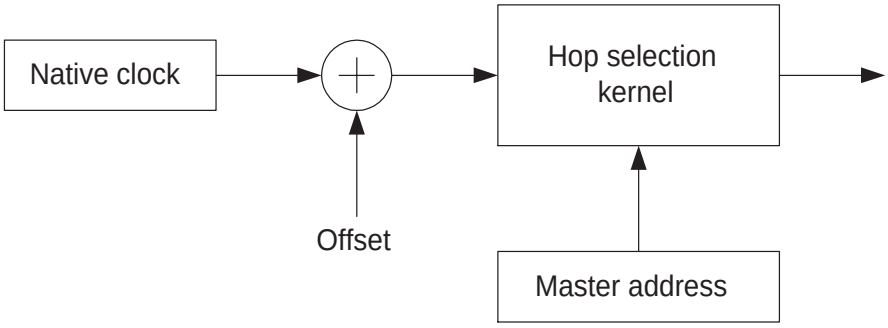

Figure 2.2: Hop frequency selection

In Bluetooth a time division multiplex (TDD) scheme is used where the master and slaves alternatively transmit. The master starts transmission only in even and slaves only in odd numbered slots (see Figure 2.3).

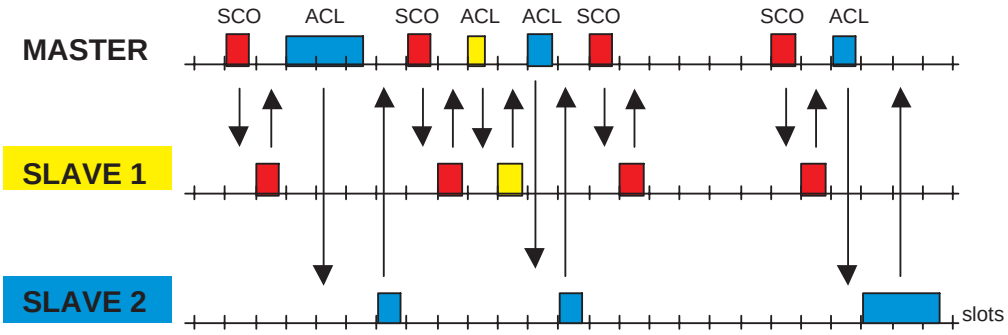

Figure 2.3: SCO and ACL links in a piconet

Between master and slave two different types of links have been defined: one of them is the synchronous connection oriented (SCO) link, which is a point-to-point link especially designed for voice traffic. SCO slots have to be pre-allocated. The master can support up to three SCO links, to the same, or to different slaves and a slave can support up to three SCO connections to the same master. The other type of link is

the asynchronous connectionless (ACL) link, typically for bursty data transmissions. ACL links are controlled by the master unit with a polling mechanism. In this case a slave is permitted to respond to an ACL packet in a slave-to-master slot, only if it is preceded by a master-to-slave ACL packet that was addressed to this unit in the slot before (see Figure 2.3) [1].

2.3 Bluetooth Packets

The length of the packets transmitted either by the master, or the node, can be one slot up to even five slots. In the case of a multi-slot packet, the RF channel must remain at the same frequency, during the whole transmission, and after finishing it, every participants in the piconet continue the hopping sequence as it is dictated by the master unit. Figure 2.4 shows the transmission of a one slot long, a three slot long and a five slot long packet.

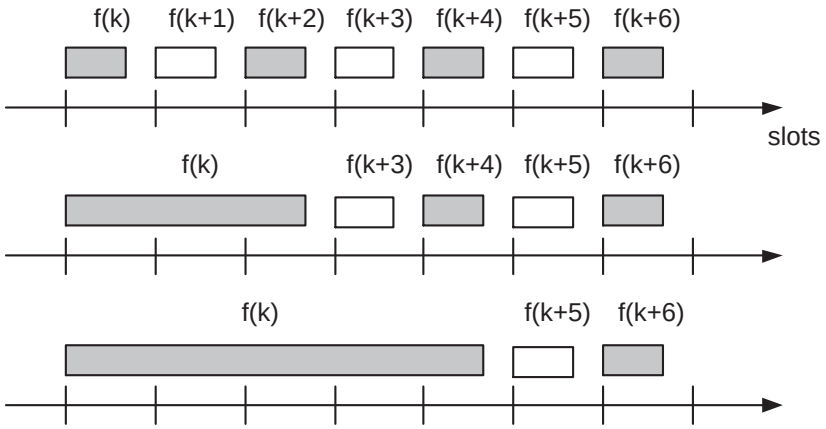

Figure 2.4: Transmission of multi-slot packets

Bluetooth packets have fixed format (see Figure 2.6). All packets sent in the same piconet are preceded by the access code. The receiver of a Bluetooth unit compares the incoming access code to the one stored in the unit itself. If they are not identical, the received packet is considered not valid on this channel, and the packet is

discarded. The access code is followed by a 54 bit long header, and after the header may come the payload.

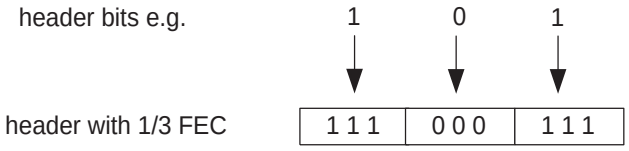

Figure 2.5: 1/3 FEC to protect the header

The access code, which plays an essential role in synchronisation and in identification, can be derived from the master identity.

Three types of access codes are used, depending on the operation mode. The channel access code (CAC) identifies the piconet, and it is included in all packets transmitted in this piconet. The device access code (DAC) is used for special signaling e.g.: paging and response to paging. The third type is the inquiry access code (IAC).

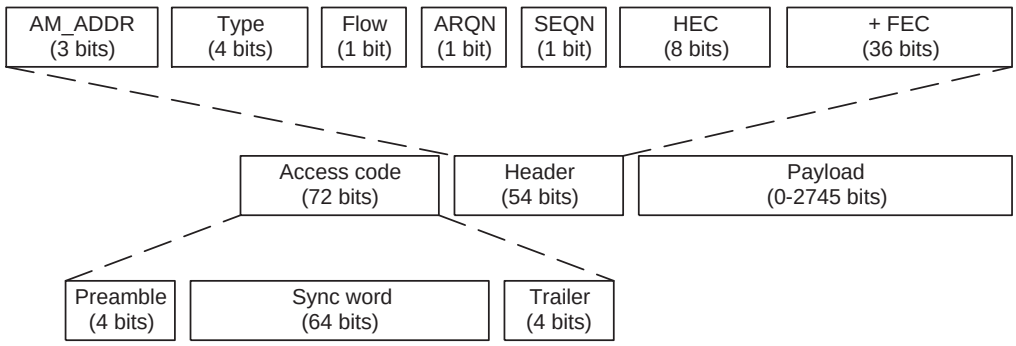

Figure 2.6: Packet format

The total header, including the HEC, consists of 18 bits, and it is encoded with a rate of 1/3 forward error correction (FEC) (see Figure 2.5), which results in a total length of 54 bits. The header contains link control information such as an active member address (AM_ADDR), packet type, flow control bits, one bit acknowledgments for the automatic retransmission query (ARQ) scheme, 1 bit sequence number and HEC. AM_ADDR is used to distinguish active members in a piconet. There are up to 7

active slaves in the piconet. To identify them, each has a temporary 3 bit address, and the all zero address is reserved for broadcast packets sent by the master.

The trailer similarly to the preamble is a fixed zero-one pattern of four symbols, either 1010 or 0101, depending on the last bit of the sync word.

2.4 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 can also be used by other devices.

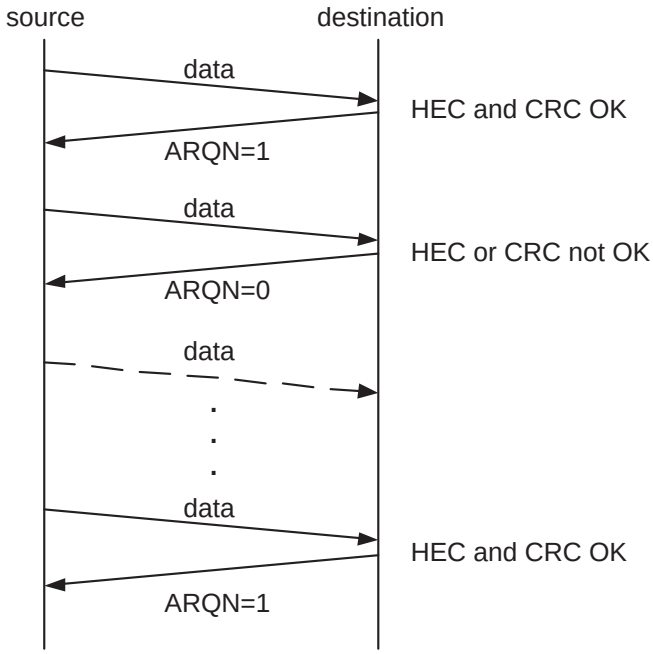

Figure 2.7: Automatic retransmission request scheme

As it has been described, the header of the packet is protected by a 1/3 forward error correction (FEC) (see Figure 2.5), since the packet header contains important information about the link.

FEC with rate 2/3 can be used for the protection of data packets. It means, that each block of 10 information bits is encoded into a 15 bit codeword. However in an error free environment this would result in unnecessary overhead that reduces throughput [15].

Therefore Bluetooth defines an automatic retransmission request (ARQ) scheme. The ARQ mechanism does not unnecessarily increase the overhead, since it retransmits packets only in case if they have been damaged. The source transmits and retransmits packets to the destination until it receives a positive acknowledgment or a timeout occurs. 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). Otherwise it sends a negative acknowledgment (NAK) packet (Figure 2.7). The slave responds in the first slave to master slot while the master responds at the next time it addresses the same slave.

This results in a fast ARQ scheme where 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.

2.5 Establishing Connections

A node can be in two major states, the Standby and the Connection state, and in seven substates: the page, the page scan, the inquiry, the inqury scan, the master response, the slave response and the inquiry response. These substates are defined to help the slaves to connect to the piconet.

2.5.1 Standby State

Standby state is the default state of a Bluetooth unit. In this state the Bluetooth unit is in low power mode, which means that only the native clock is running. A unit may leave the standby state to scan for inquiry or page messages or to page or inquiry itself. In this latter case the node is going to be the master of the piconet.

As it is mentioned in the Introduction in order to establish new connections the inquiry and the page procedure are used. The Inquiry procedure is used to discover other Bluetooth devices in range, and it is optional, while the page procedure is to establish an actual connection and it is compulsory in Bluetooth (see Figure 2.8).

Figure 2.8: Connection-establishment procedures

2.5.2 Inquiry Procedure

The inquiry procedure is used when the destination device address is unknown to the source node or when a node wants to discover other Bluetooth devices in range.

Inquiry Substate

A unit which gathers the Bluetooth device addresses and the clocks of all the units that want to be discovered enters the inquiry substate. It means that it continuously transmits the inquiry message at different hop frequencies. This message does not contain any information about the source node. It only consists of the inquiry access code which can be both the general inquiry access code (GIAC) that involves all the nodes within range or the dedicated inquiry access code (DIAC) that only refers to

nodes with special characteristics.

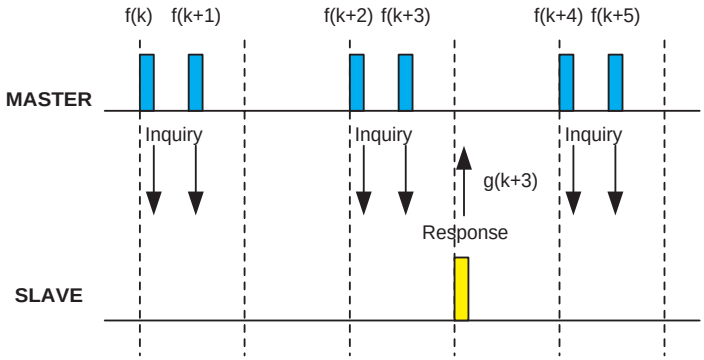

Figure 2.9: Messaging in inquiry and in inquiry scan substate

Since the inquiry message packet is very short - 68 bit long - the hop rate can be increased from 1600 hop/s to 3200 hop/s. So in a single slot the discovering unit transmits this access code twice on two different frequencies. In the next slot it listens to answers also on two different frequencies according to the inquiry response hop sequence. The discovering unit keeps transmitting this inquiry message in every even numbered slots and between two transmissions it listens for answers until it decides to have enough responses or a timeout has been reached.

Both the inquiry and the inquiry response hop sequence are derived from the inquiry access code and from the native clock of the discovering unit. The inquiry hopping sequence is divided into two trains: A and B both containing 16 frequencies. A single train must be repeated for at least $N_{inquiry}$ times before the other train can be used. The typical value of $N_{inquiry}$ is 256. In order to collect all the responses at least 3 train switches have to be done, consequently an inquiry procedure may last for 10.24s.

Inquiry Scan Substate

A unit that wants to be discovered regularly enters the inquiry scan substate. In this substate a unit scans for the inquiry access code for $T_{w\_inquiry\_scan}$ on a single hop frequency. This $T_{w\_inquiry\_scan}$ has to be long enough to completely scan for 16 frequencies. That means it has to last at least 16 slots. In every 2048 slots or in every 1.28s the frequency of the scanning is changed. If an inquiry message is received during this $T_{w\_inquiry\_scan}$ period of time, the unit enters the inquiry response substate.

Inquiry Response Substate

When an inquiry message is received (see point 1 in Figure 2.10), it has to be answered with a message that contains all the necessary information about the node itself. To eliminate the contention between the nodes that wake up at the same time every unit has to follow a contention resolution protocol. According to this protocol a unit generates a random number between 0 and 1023. During the random number slots the node may return to the standby or the connection state. So during this period of time this unit can send or receive data. After that it reenters the inquiry scan substate (see point 2 in Figure 2.10). Receiving the first inquiry message (see point 3 in Figure 2.10) the node enters the inquiry response substate and transmits its answer back (see point 4 in Figure 2.10) and starts to scan for new messages in inquiry scan substate, where receiving another message it repeats the protocol described above. If the node does not receive anything for a timeout it returns to the standby or to the connection state.

The inquiry and the inquiry scan substates can be both entered from the standby and the connection state as well. The difference lies in the fact that in the standby state a unit can use all its capacity to perform the inquiry procedure, while in connection

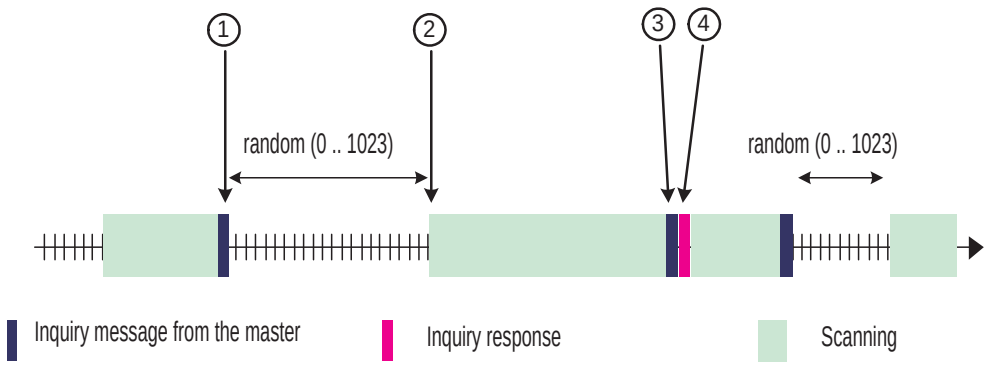

Figure 2.10: Inquiry scan substate of a slave

state it has to reserve as much capacity as needed to carry out the inquiry protocol or to scan. Namely the ACL links can be interrupted by the inquiry or the inquiry scan substate, but SCO links have higher priority than the inquiry or the inquiry scan substate. That is, a voice connection can not be interrupted, while an asynchronous data connection does not block the inquiry procedure.

2.5.3 Page Procedure

With the page procedure an actual connection can be established, if the destination node's identity is already known. This procedure is mandatory in Bluetooth [1]. The page substate is used by the master to activate a slave, which periodically wakes up for a page scan substate. In a page scan substate a unit listens to its own device access code (DAC) at a single hop frequency for a $T_{w\_page\_scan}$. If a node receives a message with its own device address, it enters the slave response substate, and transmits a response in the next slot. This response consists of only the device address of the unit itself. After the master receives this response, it enters the master response substate and sends a packet that includes all the information that is needed to establish the connection. It contains the Bluetooth device address

(BD_ADDR) of the master and the clock of the master.

Table 2.2 shows the typical times associated with connection establishment [2]. The values referring to the inquiry procedure are detailed and investigated both by simulations and analytically in Chapter 5.

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

Table 2.2: Time needed to establish connections

2.5.4 Connection State and Power Consumption

The master has to send packets periodically to all slaves in connection state 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 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 to packets addressed to them in the master to slave slots. 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 μs before the start of the next slot. The receiver searches for the channel access code for 20 μs (Figure 2.11). 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.

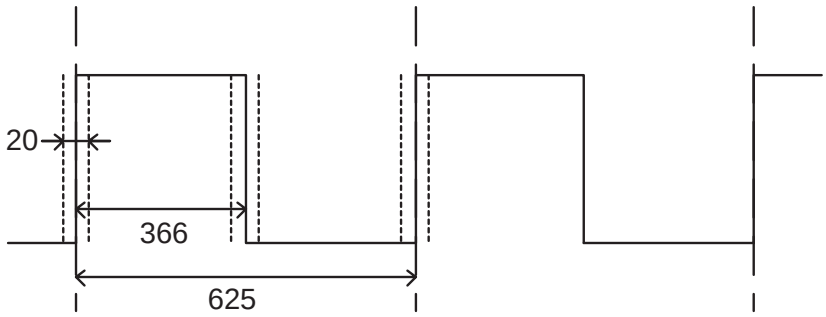

Figure 2.11: Receive window

There are further special power-saving modes to save battery when the traffic is low. These are the sniff, the hold and the park modes.

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 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.

2.6 Scatternet

In one piconet there can only be 8 active nodes altogether, but several piconets can be created at the same place with overlapping radio areas. The group of overlapping piconets is called scatternet (see Figure 2.12) [1]. Every piconet has its own hop sequence, 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.

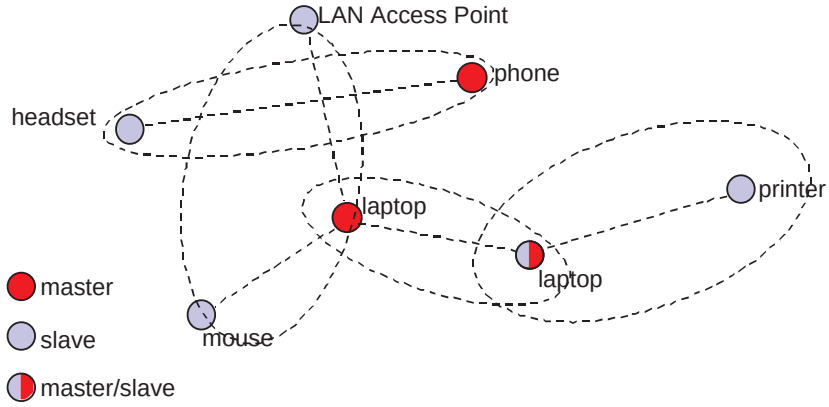

Figure 2.12: Scatternet

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 means that 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 be built with nodes which are participating in several piconets.

Chapter 3

Flexible Neighbour Discovery

In Bluetooth technology an inquiry procedure has been defined to help the nodes to discover their neighbours (nodes within radio range). I have already described this procedure in Chapter 2. While this solution is appropriate for applications where neighbour discovery is only rarely needed, it may not be optimal in a dynamically changing network, where an 10.24 s long procedure cannot be afforded. Since it has high overhead, data transmissions can be carried out only at a low rate during the procedure. Furthermore, this procedure assumes asymmetric roles meaning that one of the nodes, performing the inquiry, acts as a potential master of the piconet, while the other(s) act as slaves, performing the inquiry scan.

I investigate a new concept [5] to discover other Bluetooth devices within range. It is a flexible mechanism which can be performed with low overhead during data transmissions. The nodes performing this Flexible Neighbour Discovery procedure (FND) have a lot of freedom in deciding when and how much time they spend with the neighbour discovery. Consequently this procedure can be carried out during data transmission in a time multiplex fashion, taking the data traffic into account. The other benefit of this is that it can be performed in an environment where all of

the nodes can be peers of each other.

The main concept is that every node sends beacon messages regularly, which contains all the basic information about a node that is needed to establish a connection and to start data transmissions. If a node wants to send data to another node whose Bluetooth device address is unknown, or it just wants to update its information about one specific node or all the neighbours, it performs scanning. The scanning does not have to be continuous. It can be done for a period of time when the node does not receive or transmit data. This feature of this solution makes it possible to perform the neighbour discovery even when a node is active.

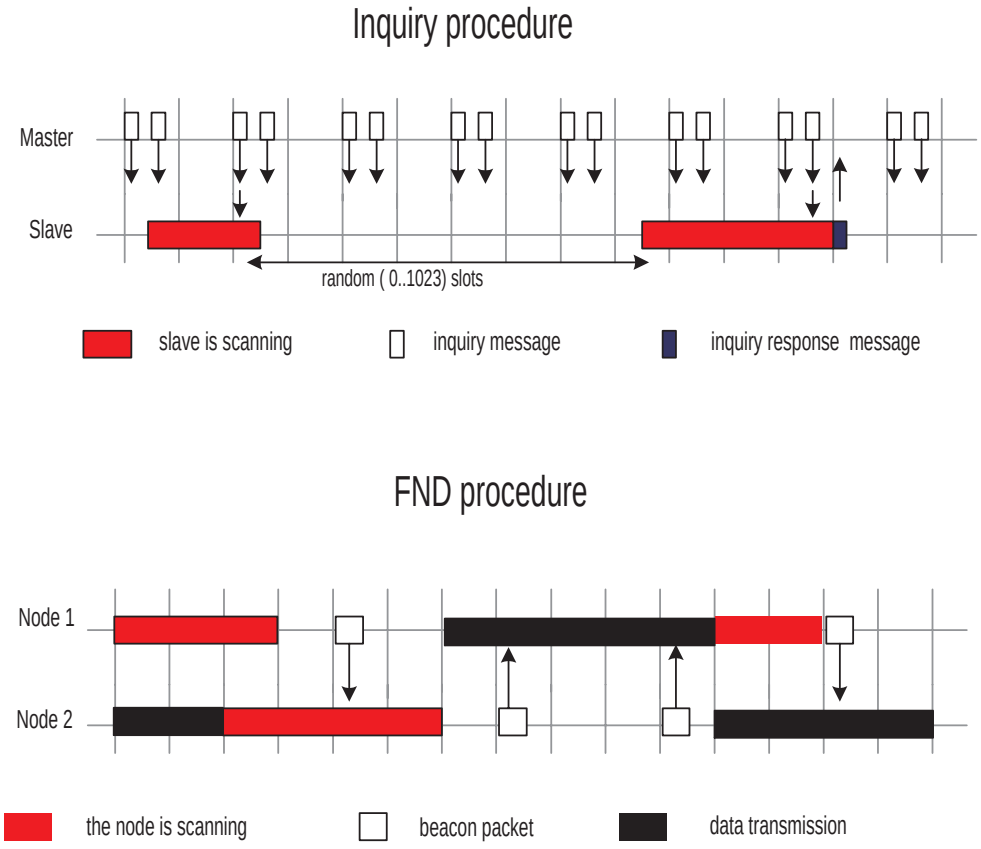

Figure 3.1: Comparison of the Bluetooth Inquiry and the FND procedure

3.1 Beacon Packets

In order to be discovered or to allow its neighbours to update their information, nodes send beacon packets at pseudo-randomly selected slots (see Figure 3.1). A beacon packet must include the MAC address and the clock of the node itself. This is especially important because from this information the neighbours can initiate data transmission to the node.

Besides neighbour discovery beacon packets can also be used to update the status information about neighbours from time to time. Therefore in this solution beacon packets have to be given the priority over baseband data packets. That means that data transmission may be interrupted in order to send the beacon. It can result in having one slot out of transmission, which can cause data or acknowledgment packet loss. To eliminate the degradation of the throughput caused by this problem, nodes should have the possibility to predict the beacon packets in advance and not to transmit during these slots. On the other hand, to eliminate the collision of beacon packets they also have to be sent pseudo–randomly.

Consequently the time of the packets have to be selected in a way that they could be predicted, but at the same time they must be also pseudo-random. There are many solutions that satisfy both requests.

In this proposal beacon periods are introduced, which are consecutive periods of equal length of T_{BCN}, where T_{BCN} is a power of two multiple of T_s (slot length). We choose one slot in every beacon period to send a beacon packet. In order to be predictable the selected slot has to be derived from the clock and the MAC address of the node itself. In the long run all the positions in a beacon period must be used with uniform distribution.

The parameter T_{BCN} is also included in the beacon packet, since it can be different from node to node, and is necessary to predict the next beacon from a peer

node. With all the information that the beacon packet contains, the beacon slot is predictable if a node needs to update its information about a specific neighbour. Since the node has to send at least one beacon in every T_{BCN} time period, even if the synchronisation of the nodes is not accurate, an interval within a beacon packet from a neighbour may arrive can be predicted.

The beacon frequency must not use the hopping sequence of the piconet because it would cause interference with data packets and would result in packet losses. It can be selected pseudo randomly and calculated from the clock and the MAC address of the node itself and it is one of the N_{BCN} frequencies. N_{BCN} is the number of frequencies used for beacon packets, and its typical values are 32 where there are 79 hop carriers and 16 where there are only 23 hop carriers.

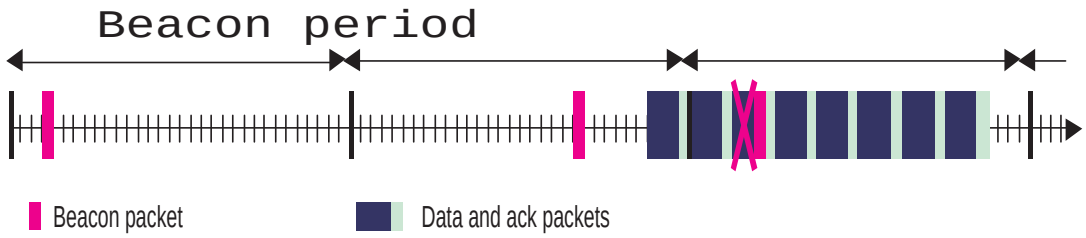

Figure 3.2: Timing of the beacon packets

Figure 3.2 illustrates the timing of the beacon periods and beacon packets of a node. The Figure also shows data transmission, and a case where a beacon packet must be sent during receiving data packets from another node. In this case one packet is lost.

3.2 Scanning

To discover its neighbours or to update the status information about its neighbours a node performs scan (see Figure 3.1). It means that for a period of time this node

does not send or receive data packets, just listens for beacon packets. The length and the timing of the scan periods are not fixed. The more often and the longer the node scans, the quicker it can discover its neighbours.

The frequency used for scanning is also not defined. In an environment where all the beacon frequencies are used with the same probability, the scanning frequency does not significantly affect the discovery procedure.

The disappearance of a neighbour can also be noticed by this procedure. If a node does not receive any beacon packet from a certain node for a given amount of time, then this node can be considered as to be moved away or turned off.

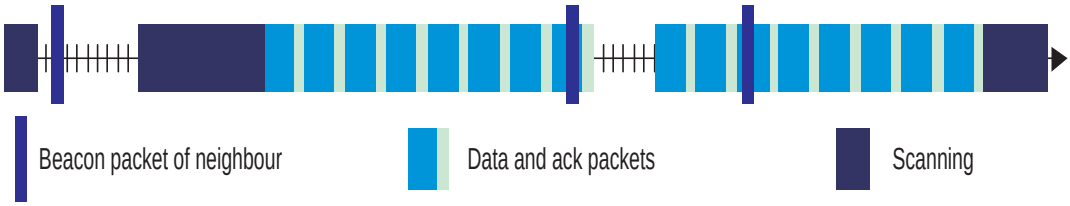

Figure 3.3: Timing of the scanning

Figure 3.3 shows a case when a node performs scanning meanwhile in a time multiplex fashion it transmits or receives data packets. The figure also shows the beacon packets of a neighbour. In this case the beacon slot has not coincided with the scanning, that means, this node has not discovered this neighbour.

3.3 Comparison to the Inquiry Procedure

The first essential difference between the two solutions lies in the fact that while in the inquiry procedure a node that wants to update the information about its neighbours sends inquiry messages, in this solution those nodes send packets who

want to be discovered, and performing neighbour discovery means that this node scans for these beacon packets (see Figure 3.1).

The second difference is while in inquiry the nodes perform asymmetric roles, the FND algorithm can be carried out in an environment where peer nodes are communicating (see Figure 3.1).

The third distinction is that although this neighbour discovery does not guarantee 100% probability of discovery in a predetermined time interval, it is a very flexible mechanism, and the probability of discovering all the neighbours goes to 1 with the time spent with scanning unlike the inquiry procedure which guarantees 100% probability of discovery. In Chapter 5 there is a more detailed comparison between the two procedures.

Chapter 4

Simulation Based Performance Analysis

In this chapter I analyze the FND procedure, using a simulation tool and numerical analysis.

I describe how I implemented the Flexible Neighbour Discovery procedure. I show the results of numerous simulations that were carried out with different parameter settings. I analyse the effects of the different parameters on the overall performance of the FND procedure, and I also give a brief numerical analysis.

Finally I introduce two kinds of mobility models. I show that under certain parameter settings the two models are equivalent. I also show the results of simulations that were carried out in a dynamically changing system.

4.1 Simulation Environment

I implemented the FND procedure in a discrete event driven, object oriented simulator environment, called Plasma [11]. Plasma is a framework for modeling data and voice networks. It is layered, which makes it easy to model the internetworking of several networking technologies. Plasma consists of several modules which can be developed and compiled separately and from the needed modules a simulator can be linked. The layered architecture and the standardised communication interfaces make it possible that objects from different modules can interoperate with each other.

In this environment the Bluetooth protocol has also been implemented [12], in a module called Bluetooth. In this module the basic Bluetooth functions have been built up.

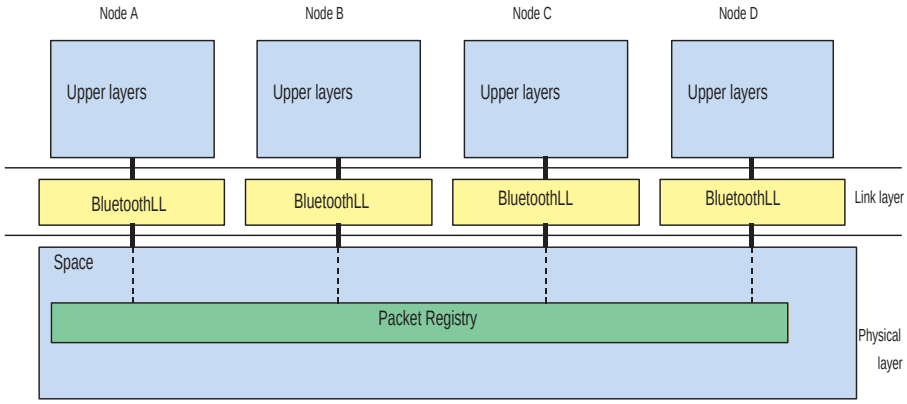

Figure 4.1: Simulator architecture

The architecture of this implementation is shown in Figure 4.1. 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

builds up connections between nodes, segments and reassembles the IP packets into one up to five slot long link layer packets, sends packets through the physical layer, and checks whether the transmission is successful. This latter task means that the node checks whether the packet frequency corresponds to the frequency the node is listening to. The Bluetooth frequency selection kernel is also implemented in the link layer (see Appendix A.1). The BluetoothLL sends the packets to the physical layer, called Space.

The physical layer contains a packet collision detector which determines the reception status of every individual packet, but the main task of the Space is to model the time spent for transactions. So it takes the responsibility to transmit the packets which it receives from the BluetoothLL layer according to the actual transmission rate of the Bluetooth network. It is also the Space which calculates the distance between nodes, so it can decide which node to transmit the packets. In physical layer model, packets are either lost due to collision, or they are delivered correctly.

In general nodes send beacon in every beacon period. To select the slot in each beacon period which satisfies the requirements that the slot have to be predictable and on the other hand it should be random, we used a pseudo random generator. To select the frequency of the beacon we used the Bluetooth inquiry frequency selection kernel, which is very similar to the Bluetooth hopping kernel selection shown in the Appendix A.1.

In our case it is the BluetoothLL that generates these beacon packets regularly. The length of the packets is 1 slot. After generating the packet the node switches to the frequency of the beacon packet and hands over the packet to the Space. After receiving it, the Space puts it into the packet registry according to the frequency of the beacon, and checks whether there is collision, that is, if there are other packets in the registry with the same timing. After a certain amount of time which is needed for the packet to reach the destination, the Space gives the packets to the nodes

which are performing scanning at that time. Then the BluetoothLLs of the nodes check whether the nodes are at the same frequency, so they can get the beacon or not.

4.1.1 The Implemented Model

I implemented two different ways of performing scanning. The first one is when a node performs scanning periodically and for the same amount of slots. The second one, when a node performs scanning randomly, but the length of the scanning is always the same (see Figure 4.2).

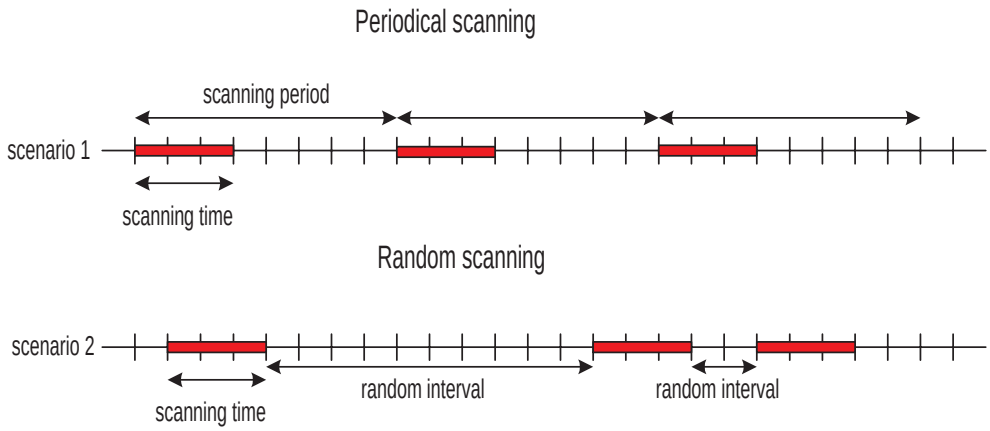

Figure 4.2: Scanning Schemes

The beacon messages are important not just because neighbours can discover each other, but also because it is useful for updating the status information about nodes within range. Therefore scan has to give priority to the beacon packets. In the implemented model instead of interrupting the scanning by a beacon packet the node that wants to perform scan investigates whether during the scanning period a beacon must be sent or not. If yes, it waits until it has been sent, and then starts the scanning, otherwise it starts scanning as it is dictated by the timing.

4.2 Simulation and Analytical Results

In this section I present the results of the simulations that were carried out to investigate the performance of the FND procedure. I studied cases when different number of nodes were present. My focus was on comparing analytical and simulation results and on identifying the effects of different parameter settings on the performance of the neighbour discovery algorithm.

4.2.1 Variable Length of the Beacon Periods

In the first experiment I studied the effects of the length of the beacon period on the time spent with scanning. Figure 4.3 shows the results of three different cases. The Figure shows the probability of the neighbour discovery as a function of the overall time spent with scanning. I carried out three different simulations with different length of beacon periods. In the first case it was 32 slot long, in the second case it was 64 slot long, while in the last case the length of the beacon period was 128 slots. In all cases there were 2 nodes in the system. One of them performed scanning with random time intervals, and the other one sent one beacon in every beacon period.

In Figure 4.3 can be seen that the probability of discovering a neighbour is an exponential function of the time spent with scanning and approaches 1. If a node sends beacon more often, for example it sends one beacon in every 32 slots, then the probability of receiving it at a random slot is higher. Therefore a node can be discovered within a certain time interval with higher probability. The results of these simulations can be explained by this fact. In Figure 4.3 the dotted grey line corresponds to the results of the case where the beacon period is 32 slots long. The black line denotes the case, where the beacon period was 64 slots long, while the black dotted line shows the results of a case where the beacon period was 128 slots long.

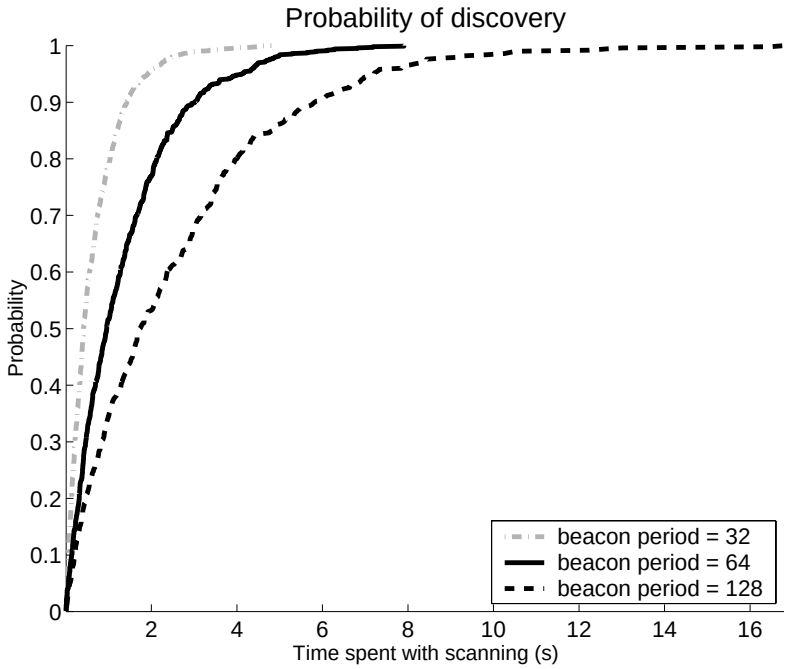

Figure 4.3: The effect of the length of the beacon period on the performance of the neighbour discovery procedure

4.2.2 Different Scanning Schemes

In the next step, I analyzed the two different kinds of scanning schemes. Both simulation setups consisted of two nodes that were within radio range. In the first case, one of the nodes was sending beacons regularly and the other node performed scanning periodically. The time between consecutive scans was a pre-defined, constant value (see Figure 4.2). In the second case the difference compared to the first simulation setup is that the time between scans was a uniformly distributed random variable (see Figure 4.2). With this simulation I wanted to investigate the difference between deterministic and random scanning periods.

The simulation results can be seen in Figure 4.4. The Figure shows the probability of the neighbour discovery as a function of the overall time spent with scanning. The grey line denotes the results of the first scenario, where one of the nodes were

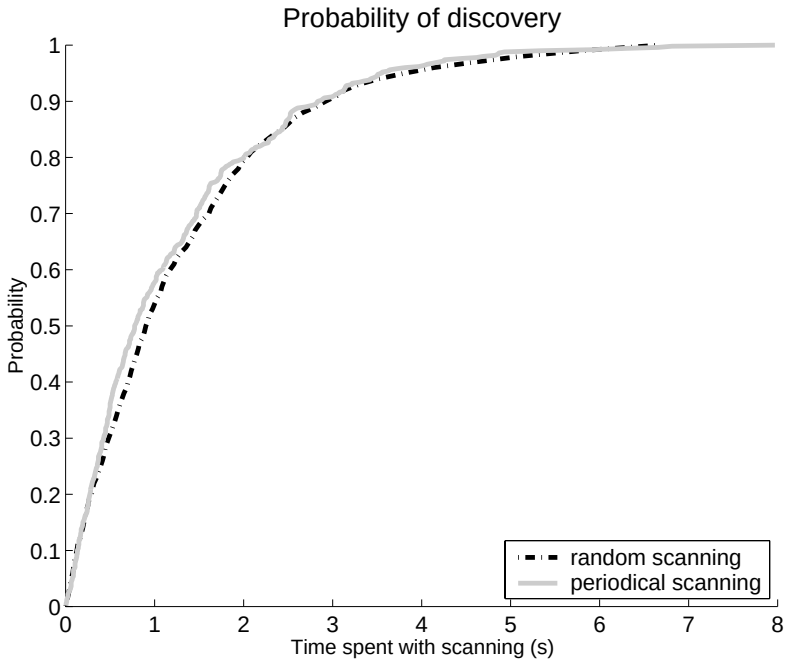

Figure 4.4: The effect of the different scanning schemes on the performance of the neighbour discovery procedure

scanning periodically: in every 100 slots for 10 slots. The black dotted line shows the results of the second scenario, where the time between consecutive scans was chosen randomly (uniformly distributed between 0 and 1000). In both cases the beacon period was 64 slots long.

It can be seen that the probability of discovering a neighbour is an exponential function of the time spent with scanning in the second scenario (dotted curve). It can also be observed that the grey line is very similar to the dotted curve. There is only a little difference between the two curves (for example between 0.5s and 2s). This can be explained by the fact that in the deterministic case the discovery probabilities in consecutive scanning periods are not independent. This effect results in some slight fluctuations in the grey curve.

4.2.3 Analytical Results

The previous simulation experiments have suggested us that the discovery probalilities follow an exponential curve. In order to prove this assumption, analytical calculations were carried out in [5] to investigate how much time is needed for discovering a neighbour with a certain probability, according to a theoretical model.

It is assumed, that a node performs scanning on a given frequency for a period of T_{SCAN}, where the value of T_{SCAN} is at least $2\mathrm{T}_S$. This requirement is necessary because the native clocks of the nodes may not be synchronised, which results in that the beginnings of the slots may not coincide in different nodes. Therefore a total time of $2\mathrm{T}_S$ is necessary to detect a beacon of length T_S. The frequency used for scanning is selected randomly with uniform distribution at the beginning of each period. A neighbour node sends a beacon in every T_{BCN} long beacon period, while the node performing the neighbour discovery has a beacon period of length T_{bcn}.

First the probability P_1 of detecting a beacon packet successfully in a single period of T_{SCAN} is determined.

$$P_1 = \left(\frac{1}{N_{BCN}}\right)\left(1 - \frac{2T_S}{T_{bcn}}\right)(1 - P_{err}), \tag{4.1}$$

where $\left(\frac{1}{N_{BCN}}\right)$ gives the probability of using the same frequency for the scanning as for the beacon. The $(1-\frac{2T_S}{T_{bcn}})$ factor takes into account that with a probability of $\frac{2T_S}{T_{bcn}}$ the beacon packet cannot be detected because the scanning node has to interrupt the scanning in order to send a beacon. $(1 - P_{err})$ gives the probability of having received the beacon successfully, if interference occurs with a probability of P_{err}. If $\mathrm{T}_{SCAN} < \mathrm{T}_{BCN}$, then the probability of having a beacon in a scan period is $\frac{T_{SCAN}}{T_{BCN}}$. Assuming that the scan periods are independent, we get that

$$P_{disc} = 1 - \left[1 - \frac{T_{SCAN}}{T_{BCN}} P_1\right]^{\frac{T_{tot}}{T_{SCAN}}} \tag{4.2}$$

where T_{tot} is the total amount of time spent with scanning. It can be shown that the limiting case of this formula, where T_{SCAN} approaches zero is

$$P_{disc} = 1 - e^{-P_1 \frac{T_{tot}}{T_{BCN}}} \tag{4.3}$$

As it can be seen Equation 4.3 does not include T_{SCAN}. It means that the overall performance of the scanning basically depends on the total time spent with scanning, and is independent of how it is divided into scan periods.

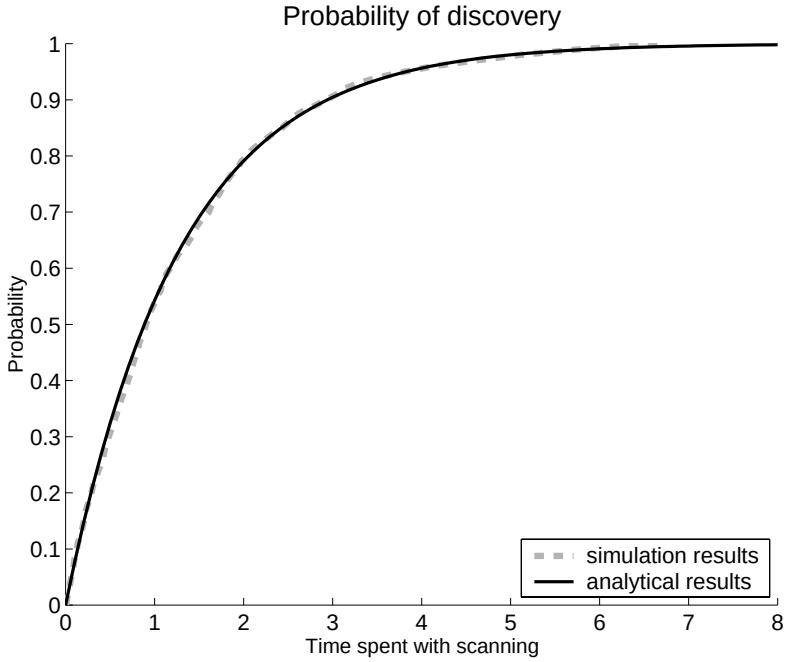

Figure 4.5: The probability of discovering a node as a function of the time spent with scanning

Figure 4.5 represents the results of the measurements where a node performed scanning randomly (uniformly distributed between 0 and 1000), compared to the curve

of the analytical model. The dotted line represents the measured values as described in the previous subsection, while the black line corresponds to the calculated values. We can see, that in the case of randomised scanning the measurements fit well to the analytical values.

It can also be seen, that 2.9s is needed to discover a neighbour with 90% probability, and another 2.9s to discover it with 99%, and in 11.6s a neighbour is discovered with a 99,99% probability. Of course these time limits refer to the time spent with scanning.

4.2.4 Variable Number of the Participating Nodes

In the last experiment I investigated the effects of the number of the nodes participating in the system. In Figure 4.6 the dotted grey line represents the original case, namely the case of the two nodes with random scanning. The plain grey line shows the results of a case with 250 nodes, where all the nodes are within radio range of each other, and the black lines correspond to the analytical results for both cases.

As we can see there is a little difference between the first two cases. It can be explained by the fact that the originally error free environment has changed when more nodes were introduced into the system. The more nodes are present in a system, the higher is the probability of the collision of beacon packets. Since the beacon period is 64 slot long in all cases, there are a lot of slots when more than one beacon is sent. If they coincide also in frequency then they will be damaged and will not be received. The packet losses result in receiving the beacons later, and that is why the continuous grey line lies below the dotted grey line.

According to the analytical results in an error free environment, the probability of receiving a single beacon is:

$$P_1 = \left(\frac{1}{N_{BCN}}\right)\left(1 - \frac{2T_S}{T_{bcn}}\right)(1 - P_{err}) \tag{4.4}$$

Where $1 - P_{err} = 0$. The probability of having collision can be calculated like :

$$P_{err} = 1 - \left(1 - \frac{1}{N_{BCN} * T_{bcn}}\right)^{(n-1)} \tag{4.5}$$

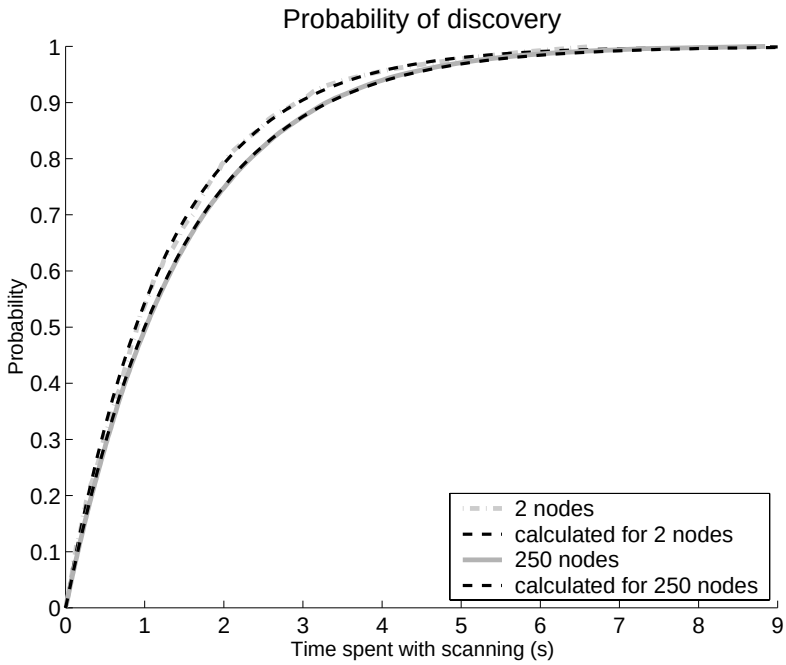

Figure 4.6: Comparing the cases where 2 nodes and where 250 nodes are present in the system

where n refers to the number of the nodes. In our case N_{BCN}=32 slots, T_{bcn}=64 slots, consequently $(1 - P_{err}) = 0.8855$. These results are also shown in Figure 4.6, and as we can see they do not differ from the simulated results.

We can see that in this case, about 3.28s is needed to discover a neighbour with 90% probability, and another 3.28s to discover it with 99%, and in 13.12s a neighbour is discovered with 99,99% probability.

4.3 Dynamically Changing System

I also investigated the performance of the neighbour discovery in a dynamically changing environment. That means in a system where the positions of the nodes are dynamically changing. Therefore I planned two different kinds of model. The first one is a so called On–Off model, while the other one is a real mobile system.

4.3.1 On–Off model

In this model the nodes do not really change their position, just with random time intervals they turn off and on. So from the point of view of one specific node (reference node) it looks as if they go out of the radio range of this node and then come back again. Figure 4.7 shows two certain moments of a simulation. We can see, that in this simulation there were 7 nodes in the system. We investigated the system from the point of view of the black node. At both moments this node can see 4 other nodes in range, but not the same nodes.

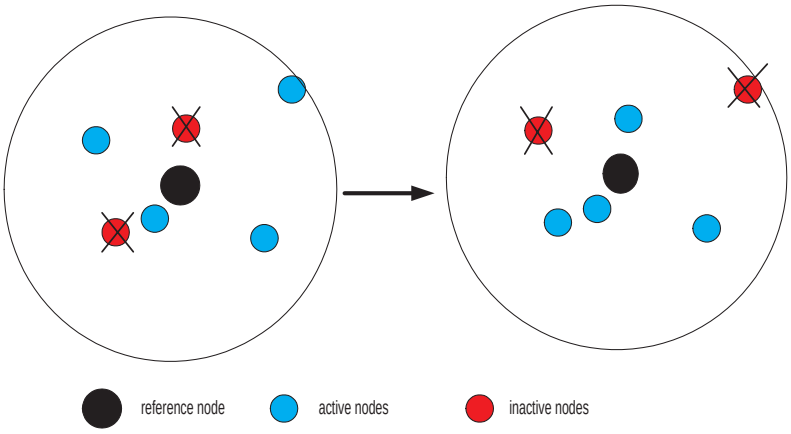

Figure 4.7: The changing environment from the point of view of a certain node

To investigate the performance of the Flexible Neighbour Discovery (FND) I measured the system in the BluetoothLL link layer and in the Space. The link layer

of the reference node recorded at certain moments how many nodes can be seen, while the Space checked the active nodes at the same moments. At the end a script evaluated the matches between the data.

Since in this experiment I wanted to study whether the reference node is able to keep up-to-date status information about its neighbours I introduced a variable which determines the life-span of the recorded entries. This variable is investigated in the next section.

Variable Entry Life-span

The results are affected by the value of the timeout after which the entries corresponding to a neighbour lose their validity.

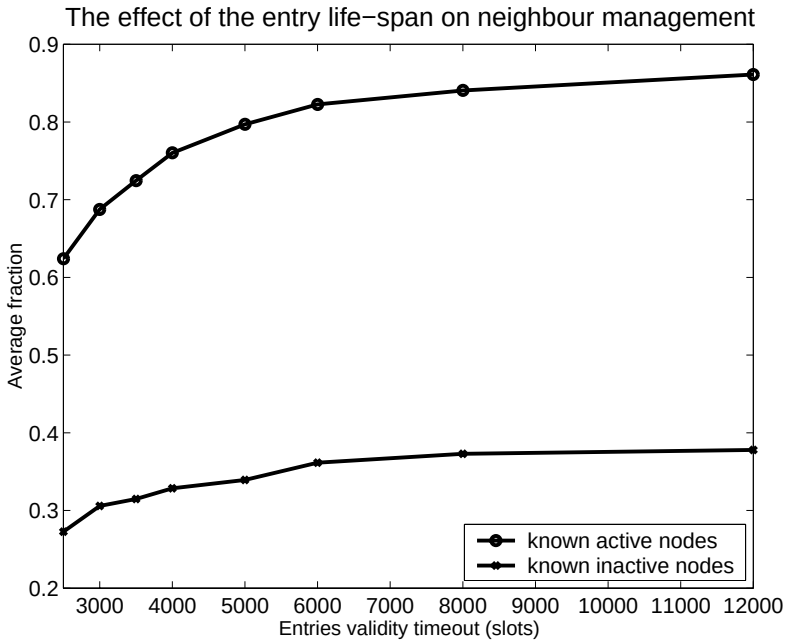

Figure 4.8: The average fraction of the known active and the known inactive nodes from the point of view of the reference node

With the first simulation I analysed the effects of the timeout mentioned above on the average number of the neighbours that the reference node can see. In the simulation

the interval within the entry about a neighbour is still alive is measured in slots spent with scanning. It means that the entry loses its validity after the reference node has spent timeout slots after it received the last beacon of this neighbour.

Figure 4.8 shows the results of several simulation with different timeouts. The timeouts range from 2500 slots to 12000 slots. In all simulation there were 10 nodes in the system. Every point is an average of 50 simulation results that lasted 1500s long. As we can see the number of the neighbours that the reference node can see increases with the life-span of the entries (see curve with circles). It can be explained that a node can turn off and then turn on again, meanwhile the reference node does not notice anything. Figure 4.8 also represents the average number of the nodes that the reference node believes to be active but they are not (see curve with x-es). As we can see it also increases with the entry life-span. Therefore the better the performance from the point of view of the active nodes, the worse the performance from the point of view of the inactive nodes.

Variable On–Off Intervals

Next I investigate the effects of the different parameter settings of this model on the overall performance of the FND procedure.

First I carried out simulations with different turning on and off intervals. The intervals within the nodes are turned off or turned on are randomly chosen with an exponential distribution. In this measurement I changed the expected value of the exponential distribution. In all cases the timeout was 4000 slots and there were 10 nodes in the system. The On–Off intervals range from 100 to 10000.

As we can see in Figure 4.9 the curve with x-es does not show any significant differences among the cases, so we can conclude, that the FND algorithm gives a very reliable solution for managing the neighbours, independently from the speed at

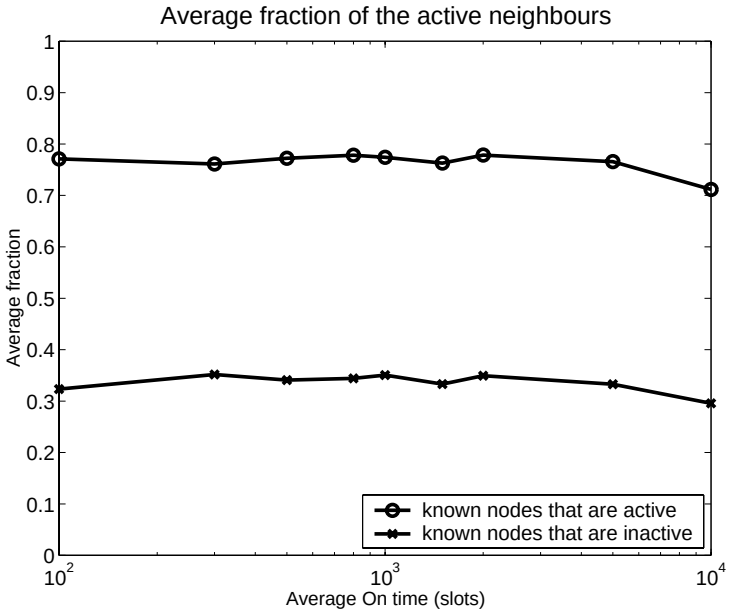

Figure 4.9: The average fraction of the known active and the known inactive nodes from the point of view of the reference node

the system changes. The measure gives another result as well which is also shown in Figure 4.9 by the curve with circles. It says, that the reference node approximately sees 3 more inactive nodes on average.

Variable Scanning Time

Finally I investigated the effect of the length of the scanning time. This is the time interval during which the node performs scanning at a single scan event. Similarly to the last case in this simulation there were 10 nodes in the system, the timeout was 4000 slots and the average expected value of the On and Off time was 1000 slots. Figure 4.10 shows that there is no significant difference among the cases, but a slight increase can be seen from 10 to 50. It can be explained by the fact that when the scanning time is shorter than the beacon period of the neighbours, then the probability of receiving a single beacon is less compared to the cases where the scanning time is longer than the beacon period. Consequently if the node is alive

and the reference node is scanning at a certain moment, in the latter cases the probability of receiving a beacon is bigger.

However the difference is very little so in this case we also can conclude, that the FND algorithm gives a reliable and flexible way of managing the neighbours independently from the scanning times.

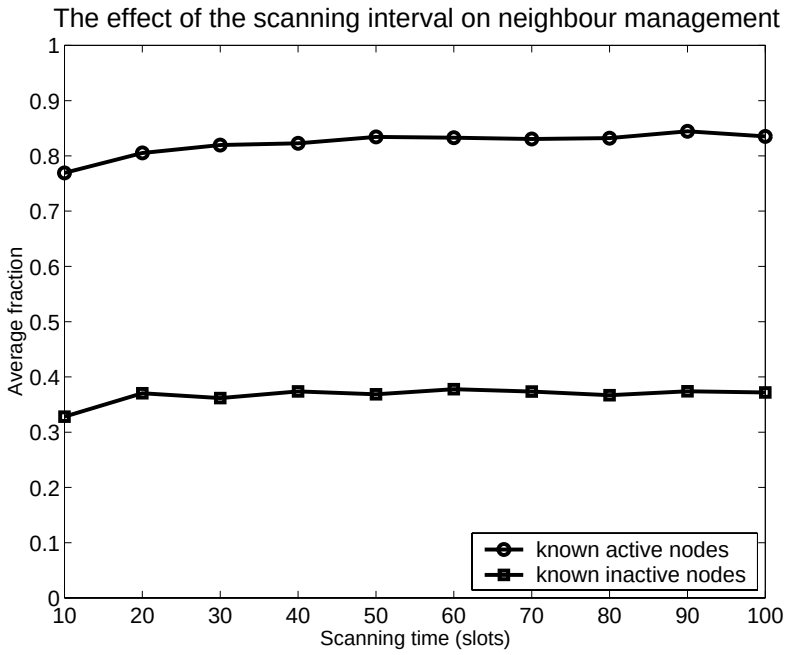

Figure 4.10: The average fraction of the neighbours that can be seen by the reference node as a function of the scanning time

4.3.2 Real Mobility Model

In this model, the position of the nodes really change from time to time. Each node moves at a certain speed in a certain direction, and they change the vector of their movement periodically.

In our Bluetooth model if two nodes are within range, then they can hear each other's transmissions, but if they are out of range they cannot hear anything. Within range

the distance between the nodes does not affect the quality of the transmissions. Since it is a discrete model this latter mobility model corresponds to the On–Off model as it can be seen in Figure 4.11. Therefore the two mobility models do not give different results.

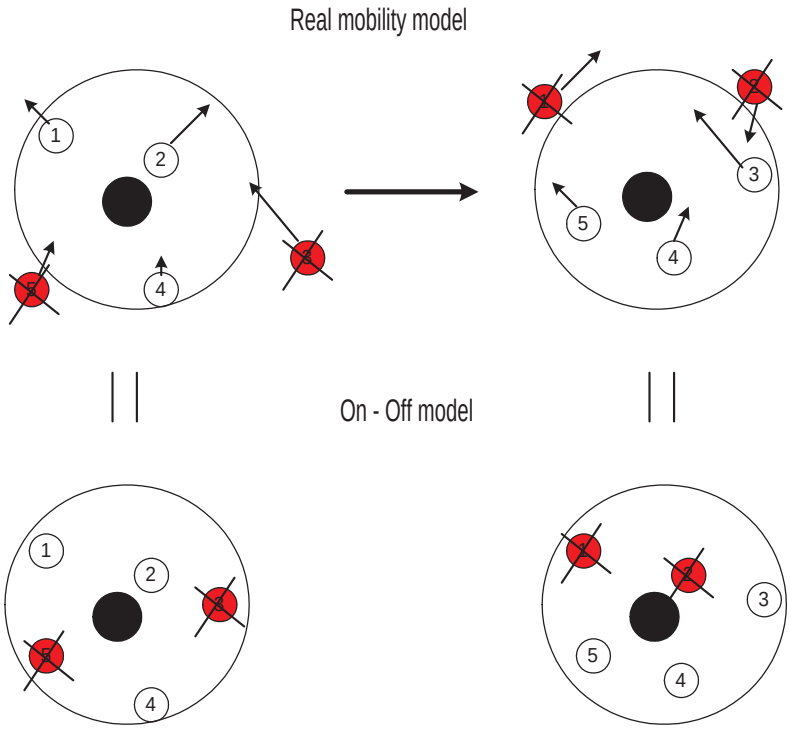

Figure 4.11: Comparing the two mobility models

Chapter 5

Comparison of the FND and the Inquiry Procedure

In this chapter I compare the two neighbour discovery procedures, namely, the Bluetooth Inquiry, and the Flexible Neighbour Discovery (FND) procedures. There is going to be comparisons from two points of view.

In the first section I investigate the two procedure in a system, where asymmetric roles are assumed, while in the second section I present the results of the simulations where symmetric roles were assumed among the nodes.

5.1 Asymmetric Roles

In this section I investigate the differences between the performance of the two neighbour discovery procedures in a scenario where asymmetric roles are assumed. In both cases there were two nodes in the system. In the inquiry scenario it means that one of the nodes was performing the inquiry while the other one woke up periodically

for a given amount of timeslots on a given frequency, and performed scanning (see Figure 5.1). I chose the wake-up period 2.56s which is the minimum requirement for Bluetooth units, and the node scanned for 16 slots long at a single frequency. In the FND scenario it means that one of the nodes sent beacon periodically, which means one in every beacon period, and the other node performed scanning at random time intervals for 10 slots (see Figure 5.1).

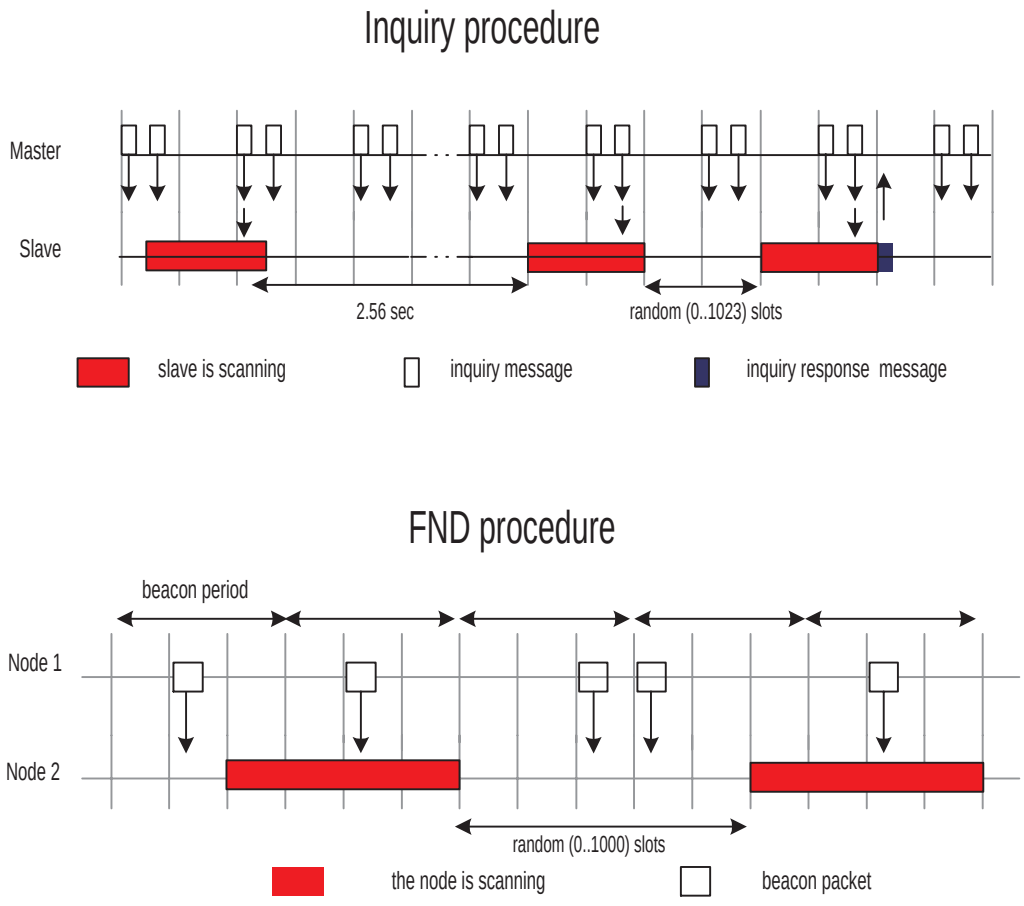

Figure 5.1: Asymmetric roles during the neighbour discovery

Figure 5.2 shows the probability of discovering a neighbour as a function of the time. The time is measured in slots spent with scanning in the case of the FND procedure, while in the case of the inquiry procedure it is measured in timeslots spent with the inquiry procedure.

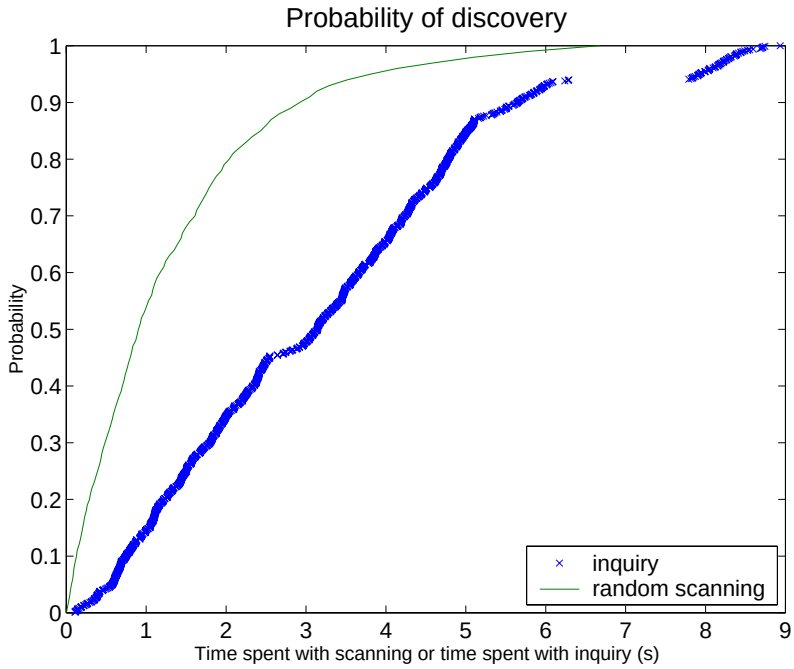

Figure 5.2: Comparing the FND and the Inquiry procedure in a system where asymmetric roles are assumed

Figure 5.2 shows the results of both cases. The curve with x-es represents the simulation results of the inquiry procedure and the black line corresponds to the measured values of the FND procedure.

In Figure 5.2 we can see that in the case of the FND procedure the node can be discovered with a probability of 90% within 2.9s, while in the case of the inquiry procedure to discover a node with this probability needs 5.65s.

In an environment that changes in every minute an inquiry procedure should be carried out in every minute, to update the information about its neighbours. If we compare the two cases from the point of view that how many percentage of the time in a minute is spent with the neighbour discovery, we also get that it is less in the case of the FND algorithm.

Assuming that the discovering node wants to discover its neighbours with a proba-

bility of 90%, then it has to perform the inquiry for 5.65s, while in the case of the FND procedure only for 2.9s. From the point of view of the other node, in the case of the inquiry procedure the scanning node spends on average 377 slots per minute with scanning, which is 0.24s, while in the case of the FND procedure the node that sends beacon regularly, uses $\frac{1}{64}$ of the time, which is 0.94s with sending beacons. To summarise it, although the FND procedure does not guarantee a certain time which is needed to discover all the neighbours, to discover the neighbours with 90% probability the FND procedure consumes 3.84s in a minute, if both roles are combined in one node. Compared to the case of the inquiry procedure where it is 5.89s.

Therefore this former procedure not only does ensure that the nodes with a probability of 90% are faster discovered but it also enables more traffic on the channels at the same time with the neighbour discovery procedure.

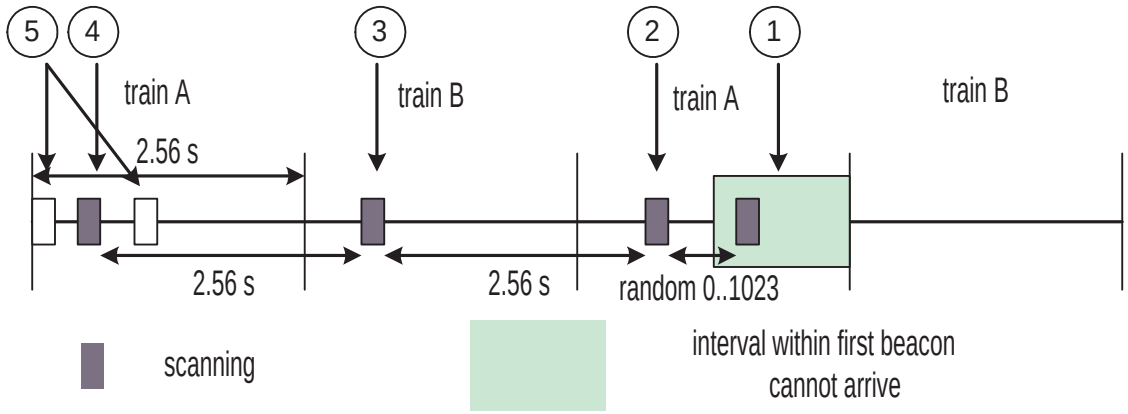

Figure 5.3: Explanation to Figure 5.2

In Figure 5.2 we can also see, that the second curve that corresponds to the inquiry procedure is not continuous. As it can be seen from 6.41s to 7.68s there is no first beacon from the nodes. The explanation is shown in Figure 5.3.

The node that performs the scanning listens at a single hop which is either in train A or in train B. Therefore a node can only send its beacon in the 6.41 to 7.68 interval, if its frequency can be found in train A. Let us assume in an indirect way that the

first beacon arrives from a certain node in this interval in the Figure 5.3. It would mean that this node performed scanning and received an inquiry message from the master right before its response (see point 1 in Figure 5.3). But it would also mean that before this scanning another scanning was performed, namely within a random interval between 0 and 1023 slots (see point 2 in Figure 5.3). It is also shown, that 2.56s before another scanning was carried out (see point 3 in Figure 5.3). This scanning must be expired, because it is sure that all the frequencies in the train B do not correspond to the one that the node is listening to. Consequently, also 2.56s before this scanning another scanning was done (see point 4 in Figure 5.3). Looking at the figure we can conclude, that during this scanning an inquiry message from the master was received, which means that either before, or after this scanning, another scanning within a time interval of (0..1023) slots was done (see point 5 in Figure 5.3). In both cases within the first quarter of the inquiry procedure a response message was sent to the master. As a result this node could not send its first beacon in the interval given above.

Dynamically Changing Network

In the second simulation I investigated the performance of the inquiry procedure in a dynamically changing network. Figure 5.4 shows the results of a simulation where there were 10 nodes in the system which turned on and turned off in an alternating fashion for random time intervals. The Figure represents the average number that the discovering node has discovered at certain moments of the simulation as a function of the average time while the nodes are turned on. Each point in the figure is an average of 50 independent measurements. In this simulation the discovering node carried out the neighbour discovery procedure in every minute and the entries about a neighbour were deleted if during the next inquiry procedure there is no updated entry.

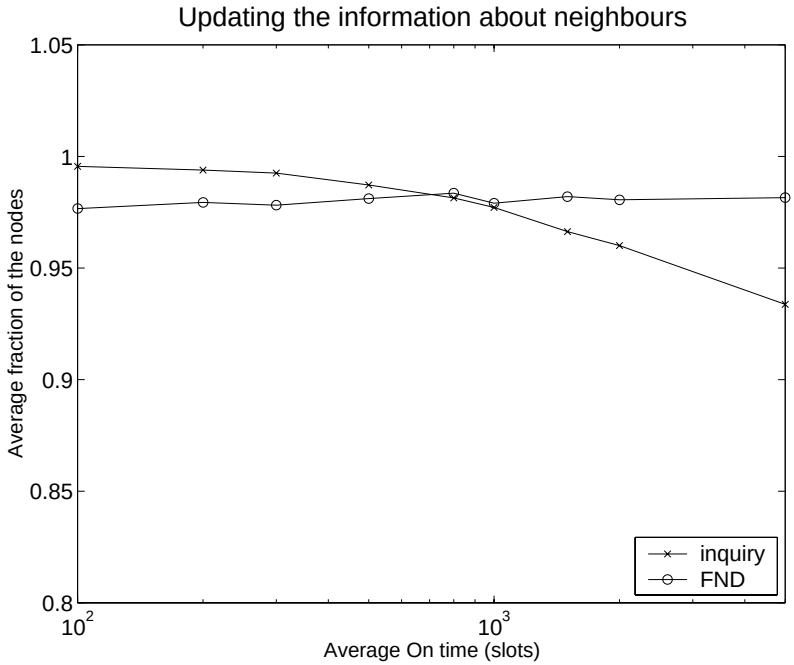

Figure 5.4: Comparing the 2 neighbour discovery procedure in a dynamically changing system

As we can see in Figure 5.4, in a fast changing system the performance of the inquiry procedure is better, than in a slowly changing system. It can be explained by the fact that if the system changes very fast then the probability of having a node discovered during the inquiry procedure is higher. While in a slowly changing system if a node is turned off during the most time of the inquiry then this node may not be discovered, therefore its entry is deleted from the neighbour registry. As a consequence this node is considered to be not alive during the whole minute. That means in order to have better results the inquiry should be carried out more often. It would result in consuming more and more capacities, only to manage the neighbours. Figure 5.4 also shows the results of the FND case. The average percentage of the nodes that are known at certain moments are lower but this curve does not show any significant difference between a slowly and a fast changing system.

5.2 Symmetric Roles

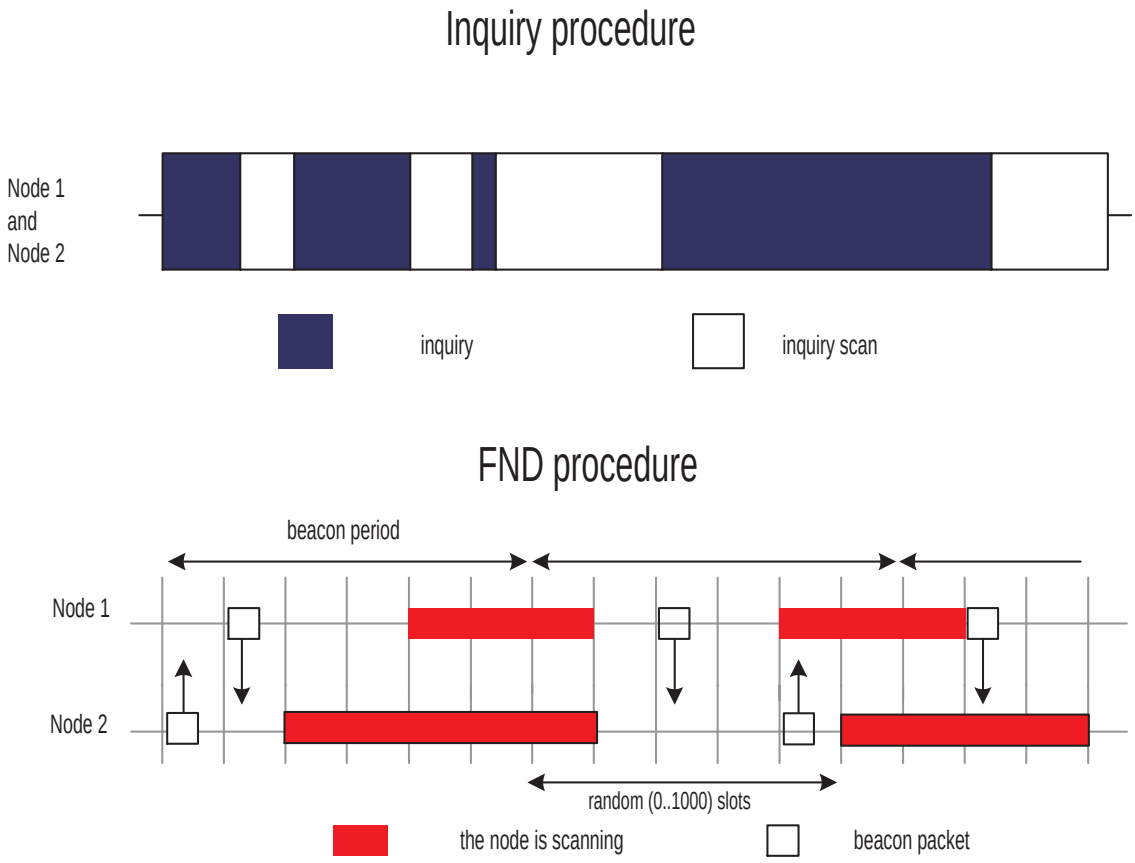

Figure 5.5: Symmetric roles during the neighbour discovery

In this section I investigate the performance of the two neighbour discovery protocols in a system where there are no predefined roles. That means in this simulation there were two nodes in the system and both of them act the same role. In the case of the inquiry procedure it means, that both of the nodes performed the inquiry and the inquiry scan substate in an alternating way (see Figure 5.5) [7], where the average value of the time spent with either inquiry or inquiry scan substate is 2000 slots which is approximately 1.2s. While in the case of the FND procedure it means, that both nodes sent beacons periodically and performed scanning at random time intervals (see Figure 5.5), where the beacon period was 64 slots and the average

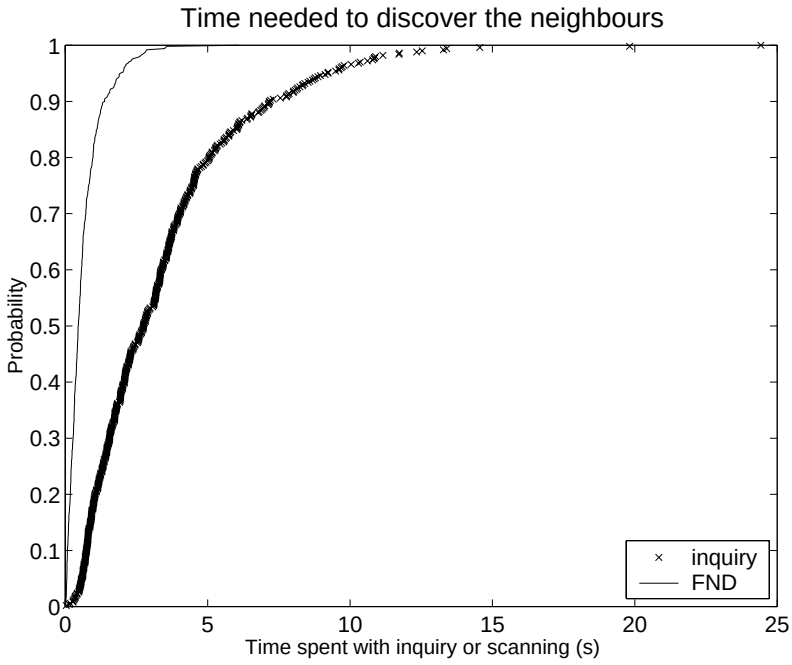

Figure 5.6: Comparing the FND and the Inquiry procedure in a system where symmetric roles are assumed

time spent between two consecutive scanning is 1000 slots.

As it is stated in [7] the connection will be established after a random delay in the case of the inquiry procedure, which is larger than the one in the asymmetric roles. The reason for that is that the nodes have to wait till they are in different substate for a given amount of timeslots that is needed to hear from each other.

The results are shown in Figure 5.6, where the curve with x-es represents the results of the inquiry procedure and the black line corresponds to the FND procedure. It can be seen that in this case the neighbours are also discovered faster with the FND procedure.

Chapter 6

Conclusion

In my thesis I have given a detailed overview of the Bluetooth Technology with special focus on the neighbour discovery method called Inquiry procedure of the Bluetooth.

I have also given an introduction about a new flexible neighbour discovery procedure (FND) which is especially designed to systems where peer nodes are communicating.

I have implemented both of the above mentioned procedures in a Bluetooth simulation tool.

I have investigated the FND procedure and have shown how the different parameter settings affect the performance of this neighbour discovery procedure. The results show that although this algorithm does not ensure a predefined time interval within the nodes can be discovered with a probability of 100%, the probability increases with the time spent with performing this procedure. It is also shown that in 2.9s the nodes can be discovered with a probability of 90%.

I have proposed two kinds of mobility models. I have supported with simulations that the speed at the system changes does not affect the performance of the FND

procedure.

I also investigated the Inquiry procedure and gave a brief comparison of the 2 procedures with simulations. The results show that both assuming asymmetric and symmetric roles among the nodes, the neighbour discovery is faster with the FND procedure, and this latter algorithm consumes much fewer capacity therefore it is more suitable in systems where a long and capacity consuming procedure can not be afforded.

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] Patrick J. Megowan, David W. Suvak, Charles D. Knutson: *IrDA Infrared Communications: An Overview*, Counterpoint Systems Foundry Inc., http://www.irda.org

[4] György Miklós: *Bluetooth: Olcsó, drótnélküli helyi összeköttetés (Bluetooth: cheap wireless local connectivity)*, Magyar Távközlés, September 2000

[5] György Miklós et al.:*Bluetooth – Best Effort*, Ericsson internal report, October 2000

[6] András Rácz, György Miklós, Ferenc Kubinszky, András Valkó: *A Pseudo Random Coordinated Scheduling Algorithm for Bluetooth Scatternets*, MobiHOC 2001

[7] Theodoros Salonidis, Pravin Bhagwat, Leandros Tassiulas, Richard LaMaire: *Distributed Topology Construction of Bluetooth Personal Area Networks*, Technical Report University of Maryland, 2001

[8] Ching Law, Kai-Yeung Siu: *A Bluetooth Scatternet Formation Algorithm*, Proceedings of the IEEE Symposium on Ad Hoc Wireless Networks 2001

[9] György Miklós, Miklós Aurél Rónai, Ferenc Kubinszky, András Rácz, Zoltán Turányi, András Valkó, Sándor Molnár: *Performance Analysis of Ad Hoc Communication Over Multiple Frequency Hopping Channels*, Ericsson internal report 2001

[10] dr. Imre Sándor: *Bevezetés a mobil számítástechnikába*, http://www.hit.bme.hu/mcl/hu/oktatas.html, 2000

[11] Zs. Haraszti, I.Dahlquist, A. Faragó, T. Henk: *PLASMA – An Integrated Tool for ATM Network Operation*, Proceedings of International Switching Symposium, 1995.

[12] Miklós Aurél Rónai: *Packet–Level Simulation of the Bluetooth Physical Layer*, Master's thesis, 2001, http://www.ronai.hu

[13] Márk Félegyházi *Development and Evaluation of a Dynamic Bluetooth Scatternet Formation Procedure*, Master's thesis, 2001, http://in3www.epfl.ch/ felegy

[14] Pablo Brenner: *A Technical Tutorial on the IEEE 802.11 Protocol*, Technical report, 1997

[15] Stefan Zürbes, Wolfgang Stahl, Kirsten Matheus, Jaap Hartsen: *Radio Network Performance of Bluetooth*, Proceedings of ICC 2000, New Orleans, 18-22 June, 2000.

Appendix A

Simulator Specification

A.1 Bluetooth Frequency Selection

Figure A.1 shows the selection kernel of the79–hop system of Bluetooth Technology. The input parameters are shown in Table A.1.

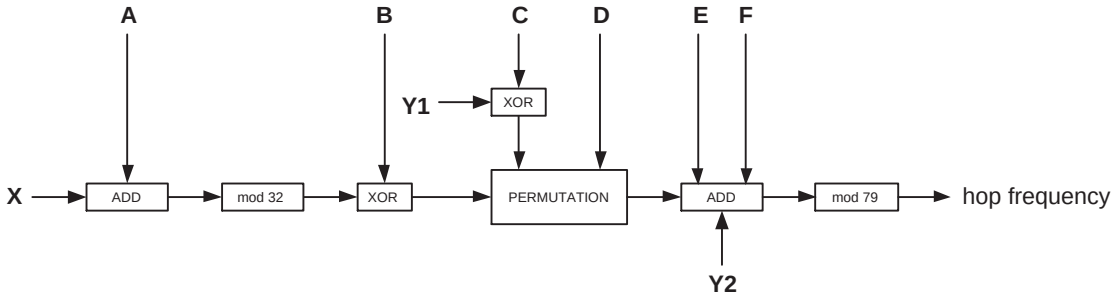

Figure A.1: Bluetooth frequency selection kernel for 79-hop systems

X	CLK_{6-2}
Y1	CLK_1
Y2	32 x 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 x CLK_{27-7} mod 79

Table A.1: Frequency selection kernel parameters

List of Abbreviations

ACK	Acknowledgment
ACL	Asynchronous ConnectionLess
AM_ADDR	Active Member Address
AP	Access Point
AR_ADDR	Access Request Address
ARQ	Automatic Retransmission reQuest
BD_ADDR	Bluetooth Device Address
CAC	Channel Access Code
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DAC	Dedicated Access Code
DIAC	Dedicated Inquiry Access Code
FEC	Forward Error Correction
FHSS	Frequency Hopping Spread Spectrum
FND	Flexible Neighbour Discovery
G-FSK	Gaussian-shaped Frequency Shift Keying
GIAC	General Inquiry Access Code
HEC	Header Error Correction
IAC	Inquiry Access Code
IR	InfraRed

IrDA	Infrared Data Association
ISM	Industrial Scientific Medicine
LAN	Local Area Network
MAC	Medium Access Control
MFHC	Multiple Frequency Hopping Channels
NAK	Not Acknowledged
PCSS	Pseudo-Random Coordinated Scatternet Scheduling
PM_ADDR	Park Member Address
SCO	Synchronous Connection-Oriented
TDD	Time Division Duplex

List of Figures

List of Tables