SAPIENZA
UNIVERSITÀ DI ROMA

Facoltà di Ingegneria

Tesi di Laurea Specialistica in Ingegneria delle Telecomunicazioni

RECOGNITION OF BLUETOOTH SIGNALS BASED ON FEATURE DETECTION

Candidato

Stefano Boldrini

Relatore

Prof.ssa Maria-Gabriella Di Benedetto

Anno Accademico 2009/2010

INDEX

CHAPTER 1
PURPOSE OF THE WORK

1.1: The cognitive radio

In the recent years the number of devices that use a wireless technology to communicate is increasing with a high rate. The motivation of this increase is probably the fact these devices guarantee (quite) the same communication performances of wired-devices, but with much more comfort and ease of use.
In fact every new communication device available on the market, such as laptops, netbooks and cellular phones, uses at least one (but frequently more) technology to interconnect to the rest of the world. For example the new cellular phones use, besides the UMTS technology, other technologies such as Wi-Fi and Bluetooth.

The enormous diffusion of such devices has emphasized a new problem: the use of the frequency band.
The frequency band is limited, and it is therefore a precious resource that must be managed in an intelligent and rational way. If there is any waste, this waste must be avoided.

The so called cognitive radio can have an important role in this context.
The cognitive radio is a wireless communication device that can adapt its transmission and reception paramaters according to the environment in which it is set in. In such way it can communicate efficiently, exploiting the unused resources, but avoiding interferences with other devices using different wireless technologies in that environment [1].
The idea of cognitive radio was first presented officially in an article by Joseph Mitola III and Gerald Q. Maguire in 1999.

As said before, the transmission and reception parameters of the cognitive radio are adapted; this means they are changed after monitoring the external radio environment, and they assume values that do not have a negative effect on the monitored environment.
In this way the cognitive radio can be seen as a "black-box" that automatically responds to the user requirements and to the network environment situation.

A distinction of two main types of cognitive radio can be done, according to the set of parameters taken into account for the decision of the transmission and reception changes:

- Full cognitive radio: in which every possible parameter observable by a wireless device is taken into account
- Spectrum sensing cognitive radio: in which only the radio frequency spectrum is taken into account

A cognitive radio has four main functions:

1. Spectrum sensing: it has to detect the unused spectrum, to find the "spectrum holes", that it can exploit to communicate
2. Spectrum management: it has to capture the best available spectrum to meet user communication requirements
3. Spectrum mobility: it has to use the spectrum in a dynamic way, because the environment situation can change in every moment and it has to automatically adapt to it
4. Spectrum sharing: it has to provide a fair spectrum scheduling method, taking into account other users in the same environment; this function is similar to the one done by the MAC (Medium Access Control) layer in existing systems

In a wider sense the cognitive radio is considered a device that can also learn from the environment, "taking decisions" according to the present condition and remembering past actions.

1.2: ISM bands

It has been said that the cognitive radio can adapt its parameters according to the environment. The environment taken into account in this work is the part of the electromagnetic spectrum defined as “ISM bands”.

The Industrial, Scientific and Medical (ISM) bands are bands of the electromagnetic spectrum that were originally reserved internationally for the use of radio-frequency electromagnetic fields for industrial, scientific and medical purposes [2].
For this reason these bands are unlicensed, and are now mostly used for non-commercial radio applications.

They were reserved by the International Telecommunication Union (ITU) in the ITU-R Radio Regulations 5.138, 5.150 and 5.280.
Despite this, the use of these bands can be different in every country because of specific national laws and regulations.

Because of this unlicensed use, communication devices using the ISM bands must tolerate any interference from other ISM equipments. For this reason, there is a strong limitation on the EIRP (Equivalent Isotropic Radiated Power). This limitation is necessary, otherwise there could be radio interferences in these bands due to very far devices, and this could prevent the use of any lower power device.

Actually the most used ISM bands are the following:

- 2.4 GHz ISM band: it goes from 2.4 GHz to 2.4835 GHz
- 5.8 GHz ISM band: it goes from 5.725 GHz to 5.85 GHz

A lot of frequently used technologies operate in the ISM bands.
For example Wi-Fi (IEEE 802.11 b/g), Bluetooth (IEEE 802.15.1), ZigBee (IEEE 802.15.4), wireless mice and keyboards and wireless closed-circuit TVs use the 2.4 GHz band. The IEEE 802.11 a uses the 5.8 GHz band, while the IEEE 802.11 n uses both 2.4 GHz and 5.8 GHz bands. The microwave oven also uses the 2.4 GHz band.

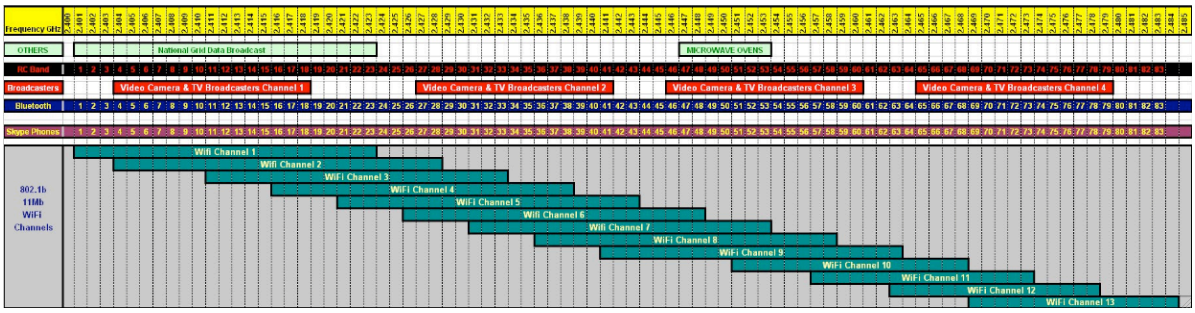

Figure 1.1 – Technologies operating in the 2.4 GHz ISM band

A cognitive radio operating in the ISM bands could discover if these bands are fully occupied or if there is any chance to exploit unused spectrum holes.

Regulatory bodies in various countries found that most of the radio frequency spectrum was inefficiently used. Independent studies performed in some countries, confirmed that observation, and concluded that spectrum utilization depends strongly on time and place.
This is more true for the ISM bands, because they are unlicensed, and anyone can use a device operating in that bands.

1.3: Discrimination among different technologies: the features approach

In this work the 2.4 GHz ISM band has been considered, and the technologies that operate in it.
As said before, there are a lot of technologies in this band, such as Wi-Fi, Bluetooth, wireless mice and keyboards and microwave ovens (just to mention the most widespread among them).

In this context, the problem a cognitive radio should be able to solve, is to know what there is in the frequency environment, which technologies are actually used in a certain moment and in a certain place.

In other words this is a problem of wireless network discovery.

Knowing which technologies are active in a certain moment and place, solving this problem, is very important, because it can make possible a further step: taking advantage of this knowledge by exploiting the unused available resources.

For this reason the primary step is the network discovery.
Considering the 2.4 GHz ISM band, this means a problem of discrimination among the technologies that operate in that band, mentioned above.

Other works propose different approaches to solve this problem.
This work's proposal is to solve the problem of discrimination among the wireless technologies working in the 2.4 GHz ISM band using a features approach.
The final goal is to use high layer features (i.e. higher than the physical layer, for example characteristics of the MAC layer of the different technologies), if it is possible, because it can lead to simpler algorithms of recognition and because it allows the use of a generic device (an energy detector, for example), that can be integrated in the cognitive radio.
If this can not be done, i.e. high layer features can not be extracted, physical features must be used.

In order to do this, a study of the standards that define each technology is necessary, to fully understand how these technologies behave, both in MAC layer and PHY layer, and to exploit the characterizing features of every technology.

In this work, the case of the Bluetooth technology (IEEE 802.15.1 [4]) is considered. This technology is described in its principal aspects, as the standard defines it.
Then Bluetooth packets are simulated, and some characterizing features are identified and extracted. Through these features the Bluetooth technology is recognized, and the results are presented.

A further step is presented: the real Bluetooth traffic capture, to move from a simulated environment to a real one.

CHAPTER 2
BLUETOOTH TECHNOLOGY

2.1: Overview and WPANs

The Bluetooth technology is widely used for Wireless Personal Area Networks (WPANs).
The scope of WPANs is to convey information between devices over short distances.

From the name we can understand the basic purposes and characteristics of these kind of networks. They are “Wireless”, that means they involve little or no infrastructure; and they are “Personal”, that involves a restricted number of devices, an intimate group, with no direct connectivity to the rest of the world.
The latter characteristic makes WPANs very different from WLANs (Wireless Local Area Networks), even though they may coexist in the same area and in the same frequencies.

Thanks to these “limited” purposes, small, power-efficient and inexpensive solutions are possible, and so the Bluetooth technology can be implemented in a wide range of devices. The IEEE Standard itself, that defines the Bluetooth technology characteristics, mentions these key features: robustness, low power and low cost.

In fact, nowadays quite every cellular phone and netbook, just to mention the smallest devices, has an integrated Bluetooth transceiver; and the external transceveirs are smaller than a coin.

2.2: General description of the Bluetooth technology

The Bluetooth technology is described in the IEEE Standard 802.15.1 – 2005 [4]. In particular, the 15.1 part describes all the specifications for the Physical Layer (PHY) and the Medium Access Control Layer (MAC).

The Bluetooth version described in this standard is known as "Bluetooth 1.2", and is an enhanced version of the "Bluetooth 1.1", described in the IEEE Standard 802.15.1 – 2002. Later other versions were defined ("Bluetooth 2.0", "Bluetooth 2.1" and "Bluetooth 3.0"), but these versions are not described in IEEE Standards; they are described in documents of the Bluetooth Special Interest Group (SIG).
In this work we refer to the IEEE Standard 802.15.1 – 2005.

2.2.1: Topology: piconet and scatternet

The context of every link between two or more Bluetooth devices is the piconet.
A piconet is composed by 2 to 8 devices that occupy the same physical channel, that is to say they are synchronized to a common clock and hopping sequence.

In every piconet only one device has the role of master, while the others assume the role of slaves. The master is the centre of this kind of topology; this means that every slave in the piconet communicates directly only with the master. In case a slave wants to communicate with another slave, it sends the packets it wants to transmit to the master, who "redirects" them to the receiver slave.
In other words, each link between devices passses through the master, the "principal" device of the piconet.

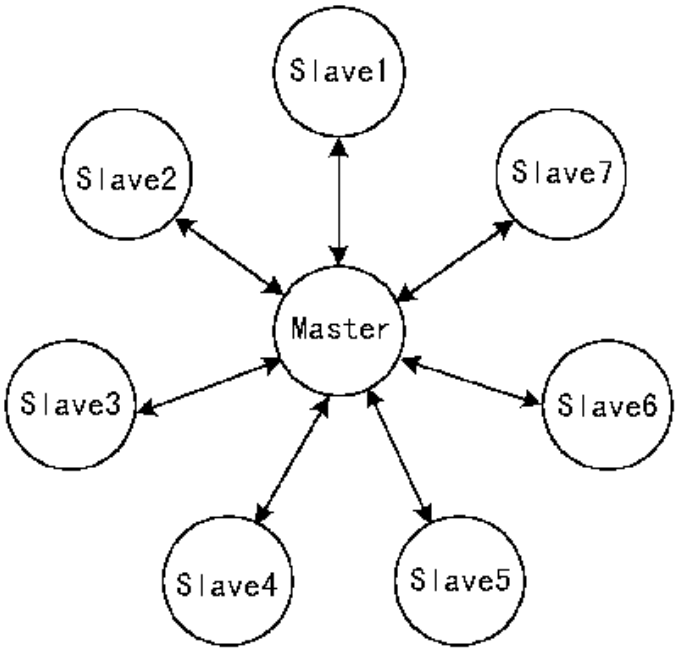

Figure 2.1 – A piconet composed by 1 master and 7 slaves

Usually the master is the device who begins the communications, but there may be a role change, where a slave of the piconet assumes the role of master (and, of course, the previous master becomes a slave, because there may be only one master in a piconet).

As said before, the maximum number of devices in a piconet is 8: 1 master and 7 slaves.
In some cases more devices could be interconneted, but in these cases a piconet is not sufficient. So the topology becomes a little more complex, and two (or more) piconets can be interconnected to form a scatternet.

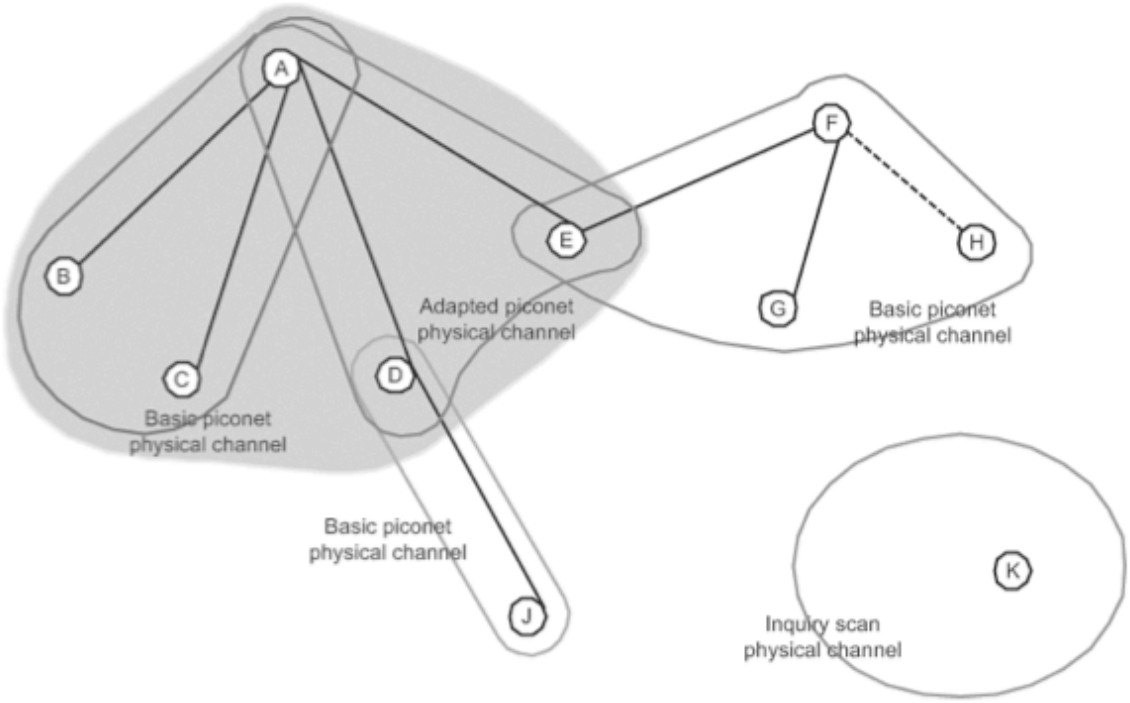

Figure 2.2 – More piconets form a scatternet

A scatternet is made of two (or more) piconets, linked together by some devices that belong to both the piconets. These piconets are linked, but not synchronized: every piconet has its own clock and hopping sequence, that means its own physical channel, different from the others. In fact they are independent.

It is important to say that, even though piconets are interconnected in a scatternet, it does not imply any network routing capability. This functionality may be eventually implemented in higher layer protocols, but it's not described in the IEEE Standard 802.15.1 - 2005.

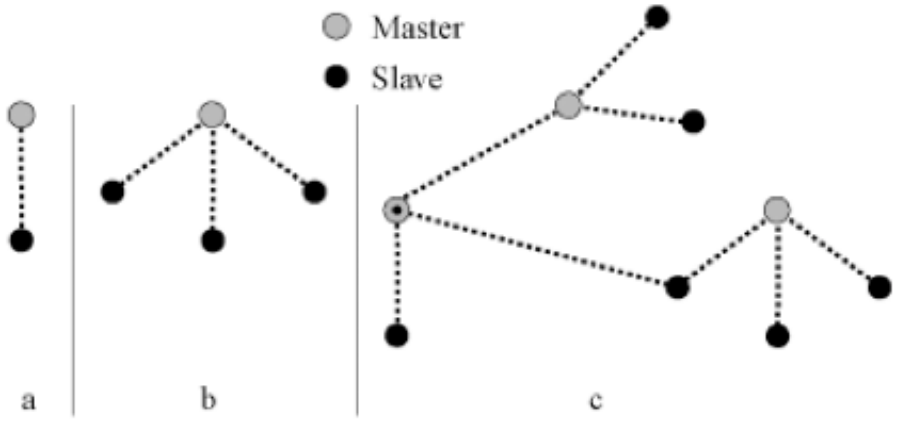

Figure 2.3 – Master and slave role in piconets and scatternets

The devices that are part of more than one piconet, can do this on a TDM basis.
A device can belong to more piconets, but can be master only in one piconet. In fact a device can be master in a piconet, and can be in the same time slave in other piconets.

2.2.2: Frequencies

The Bluetooth uses the ISM (Industrial, Scientific and Medical) unlicensed band at 2.4 GHz, that goes from 2.4 GHz to 2.4835 GHz.
In every moment the signal occupies a band of 1 MHz, but it uses the whole ISM band using a Spread Spectrum technique: the Frequency Hopping Spread Spectrum (FHSS). This technique is used to combat interference and fading.

The whole ISM band is divided into 79 channels of 1 MHz. There are also guard bands: the lower one is of 1.5 MHz, and the upper one is of 3 MHz.
So the centre frequency of the channel number "k" can be written as

$$f = 2402 + k \quad \text{MHz}$$

where k = 0, … , 78.

When two devices want to communicate, they must before agree on the hopping sequence they will follow to transmit and recevice. When they share this hopping

sequence, they can begin to send and receive the data packets, because they know in every instant in which channel to transmit and receive and which channel will be the next one.

2.2.3: The hopping sequence

In this context the hopping sequence assume a very important role: it is necessary to communicate.
The IEEE Standard 802.15.1 - 2005 defines 6 types of hopping sequence:

1. Inquiry hopping sequence
2. Inquiry response hopping sequence
3. Page hopping sequence
4. Page response hopping sequence
5. Basic channel hopping sequence
6. Adapted channel hopping sequence

Every type mentioned is strictly connected to its specific state of the Bluetooth device (states and substates are discussed later).

The first 4 types use 32 channels equally distributed over all the 79 possible channels. The inquiry response hopping sequence is in one-to-one correspondence to the current inquiry hopping sequence, and we have the same situation for the page response hopping sequence and the current page hopping sequence.
These kind of sequences are used in the inquiry, inquiry response, page and page response substates.

The basic channel hopping sequence uses all the 79 channels, has a very long period length and does not show repetitive patterns over a short time interval. This sequence is used during the connection state, that is to say when the devices have already built up a physical link and they are ready to send and receive the data packets.

The adapted channel hopping sequence is derived from the basic channel one, but may use a lower number of channels. It is only used in place of the basic channel hopping sequence.

The hopping sequence is generated from the master's device address (BD_ADDR, explained later) and its clock.

2.2.4: Power levels

The IEEE Standard 802.15.1 - 2005 divides the devices in 3 classes according to their maximum output power (P_{MAX}):

- Power class 1 P_{MAX} = 100 mW (20 dBm)
- Power class 2 P_{MAX} = 2.5 mW (4 dBm)
- Power class 3 P_{MAX} = 1 mW (0 dBm)

These powers allow the devices to cover an area with an approximate range of 100 metres (power class 1), 10 metres (power class 2) and 1 metre (power class 3).

Talking about the receiver characteristics, the IEEE Standard specifies the reference sensitivity level at -70 dBm.
The receiver must have a sensitivity below or equal to the reference level.
Another important parameter for the performance is the BER (Bit Error Rate), and the IEEE Standard specifies a reference sensitivity performance with BER = 0.1 %. From this parameter the actual sensitivity level is defined: it is the input level for which a raw BER of 0.1 % is met.

2.2.5: Modulation characteristics

The modulation technique used in the Bluetooth technology is the Gaussian Frequency Shift Keying (GFSK).
The GFSK is a kind of continuous phase frequency shift keying (CPFSK), so it is originated from the FSK (as the name says), but a gaussian function pulse shaping filter is used to reduce the transmission bandwidth.

A binary zero is represented by a negative frequency deviation, while a binary one is represented by a positive frequency deviation.
Whitout the continuous phase and the gaussian filter, the occupied bandwidth would be greater. In fact, a discontinuity in the phase, that would occur when the istantaneous frequency changes because the bits sequence passes from 0 to 1 or from 1 to 0, would cause a band enlargement; that is why the phase is kept continuous. But even if it is continuous, without the gaussian filter there could be fast frequency deviations, that would also cause a band enlargement. This filter reduces the bandwidth because it smooths the frequency deviations that represent the binary 0 and 1.

The functional blocks of a GFSK modulator are shown below [7]:

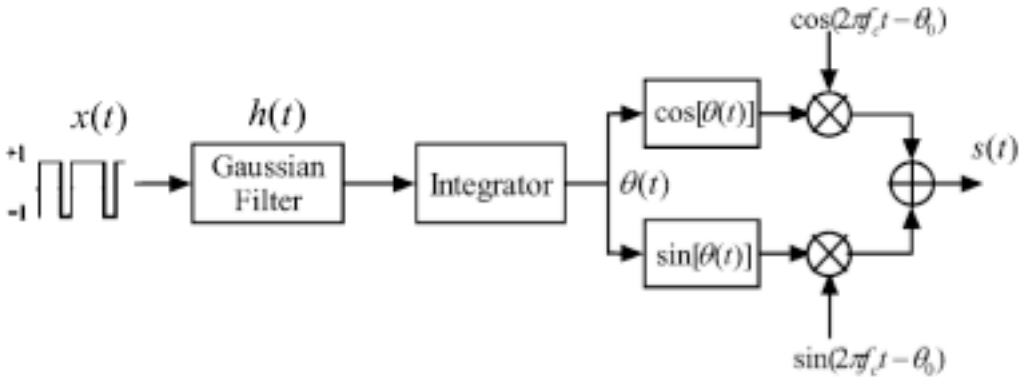

Figure 2.4 – GFSK modulator

The bit sequence is first of all translated from unipolar to bipolar. The consequent waveform is filtered through the gaussian filter and then modulated by the frequency generator, consisting of an integrator and a quadrature frequency synthesizer.

Expressing all of this in mathematical terms:

the bipolar bit sequence is: $x[n] \in \{-1,+1\}$

the input signal is therefore: $$x(t) = \sum_{n=-\infty}^{+\infty} x[n] \cdot \prod\left(\frac{t-nT}{T}\right)$$

where T is the symbol period (in this case the bit period) and a rectangular pulse function is used: $\prod\left(\frac{t}{T}\right) = \frac{1}{T}$ only for $|t| < \frac{T}{2}$ (and 0 otherwise).

The impulse response of the gaussian filter is: $$h(t) = \frac{1}{\sqrt{2\pi}\sigma T} e^{-\frac{t^2}{2\sigma^2 T^2}}$$

where: $$\sigma = \frac{\sqrt{\ln 2}}{2\pi BT}$$

B is the 3 dB bandwidth and the bandwidth–bit period product BT is 0.5 (imposed by the IEEE Standard).

After the integration: $$\vartheta(t) = 2\pi h \int_{-\infty}^{t} \sum_{n=-\infty}^{+\infty} x[n] g(\tau - nT) d\tau$$

with h modulation index (imposed in the range 0.28 – 0.35 by the IEEE Standard)

and where: $$g(t) = \frac{1}{T}\left[Q\left(\frac{t-T/2}{\sigma T}\right) - Q\left(\frac{t+T/2}{\sigma T}\right)\right]$$

with the Q-function defined as: $$Q(t) = \frac{1}{\sqrt{2\pi}} \int_t^{\infty} e^{-\tau^2/2} d\tau$$

At the end the analytical expression of the GFSK modulated signal with unit power and centre frequency f_c is:

$$s(t) = \sqrt{\frac{2}{T}} \cos[2\pi f_c t - \vartheta(t) - \vartheta_0]$$

where θ_0 is the initial phase offset.

These characteristics allow the data to be transmitted at a symbol rate of 1 Msymb/s. The IEEE Standard 802.15.1 - 2005 writes about a bitrate of 1 Mb/s, but in later versions of the standard it is specified a higher bitrate (3 Mb/s in the Bluetooth 2.0 version + Enhanced Data Rate).

2.2.6: Device address

Every Bluetooth device has a 48-bit address, called BD_ADDR. This address is unique and it is obtained from the IEEE Registration Authority.

BD_ADDR is divided into three fields:

1. LAP (Lower Address Part) 24 bits
2. UAP (Upper Address Part) 8 bits
3. NAP (Nonsignificant Address Part) 16 bits

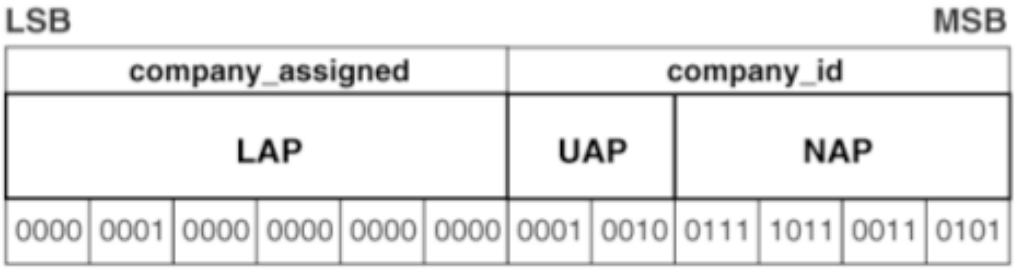

Figure 2.5 – BD_ADDR

The 24 bits that form UAP and NAP usually identify the name of the company that produces the device, while the other 24 bits of LAP are assigned by the company itself and identify the single specific device.

There are 64 LAP values that are reserved for inquiry operations (explained later): 1 for general inquiry, the other 63 for dedicated inquiry (used to find specific classes of Bluetooth devices).
These reserved LAP addresses are: 0x9E8B00 – 0x9E8B3F, where the one used for general inquiry is 0x9E8B33 (in hexadecimal notation). In these values it is intended that the Least Significant Bit (LSB) is at the rightmost position.

2.3: Clock and time slots

2.3.1: Clock

Every Bluetooth device has a clock that ticks with a period T = 312.5 μs (the rate is F = 3.2 kHz). This clock is completely independent from the time of day, because it can be turned on in any instant and initialized to any value.
The cycle of the clock must be approximately a day; this means it turns back on the current value only after about 24 hours.

In the devices the clock is usually implemented with a 28-bit counter. The Least

Significant Bit (LSB) changes his value every 312.5 μs, exactly a period T. So, with 28 bits, it has a cycle of about 23.3 hours, according to the IEEE Standard specifications.

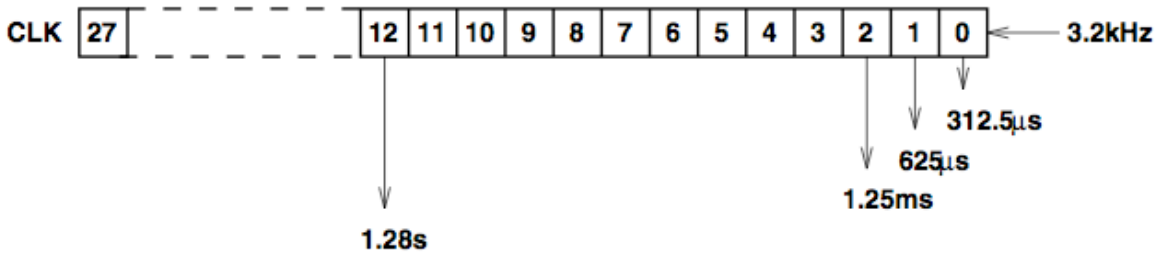

Figure 2.6 – Clock implemented with a 28-bit counter

Naturally, every device clock is free running; in other words, it has no relation with the other devices clocks.
In order to communicate, the devices must be synchronized, that means share a common clock value. Their clocks run at the same nominal rate, but there can be inaccuracies derivated from drifts.
To synchronize their clocks, the devices use offsets, from which they know how to "adjust" the clock value.

To understand better this mechanism of synchronization, we can talk about three types of clock:

1. CLKN, the native clock
2. CLK, the master clock
3. CLKE, the estimated clock

CLKN is the native clock, the clock every device has, as it is.
CLK is the clock of the master device, and it is used as reference clock for the whole piconet. Every device extracts CLK from its own CLKN adding the correct offset, that is calculated from the master's packets. Then the offset is regularly updated to keep the devices synchronized.
Naturally the master already knows the CLK: it is its own CLKN, so it has to add nothing.

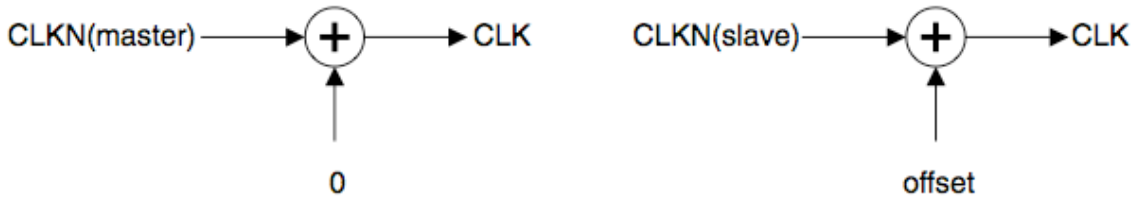

Figure 2.7 – Calculation of CLK

CLKE is an estimation of the page scanning device clock made by a paging device. (As explained better later, a page scanning device is a device in the page scan substate, and of course, a paging device is a device in the page substate). The paging device estimates an offset and adds it to its CLKN, to extract an estimated version of the page scanning device clock. It does this operation to speed up the connection establishment, that is to say to pass in the connection state.

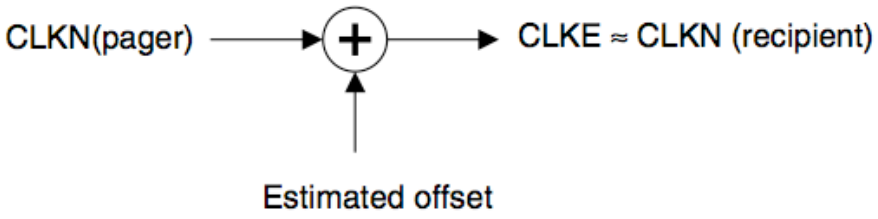

Figure 2.8 – Calculation of CLKE

2.3.2: Time slots

The master and a slave access the physical channel according to a Time Division Duplex (TDD) scheme, realizing a TDD full-duplex connection.

A physical channel is divided into time slots. Every time slot lasts 625 μs, which means two clock cycles. So we have a new time slot when the second LSB of the clock, CLK_1, changes its value.
Master and slave transmit alternatively: the master sends its packet, and the next slot is reserved for the slave's packet, and so on. Every packet starts at the beginning of the time slot, and it can last up to 5 time slots (not more). To be more precise, the

master can begin a packet transmission only in even time slots (when $CLK_1 = 0$); the slave's packets can begin only in odd time slots (when $CLK_1 = 1$).

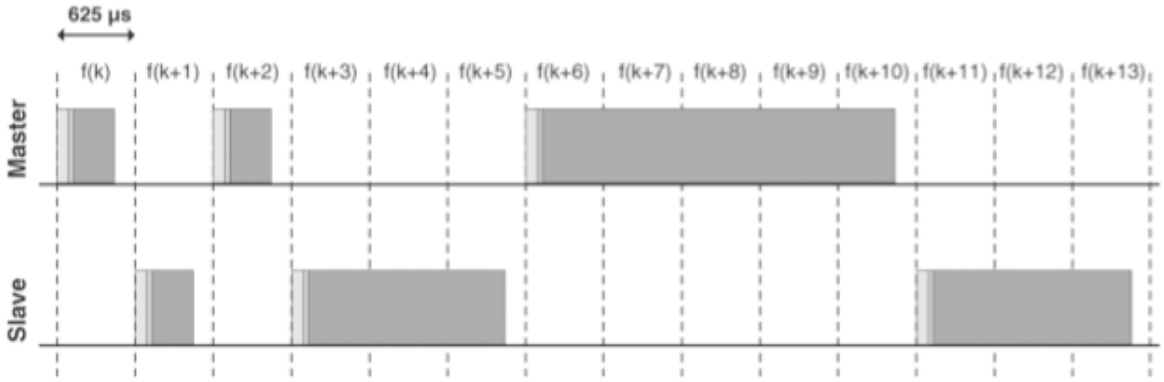

Figure 2.9 – Slot timing

When a device sends a packet, it waits for a return packet (at least an acknowledgment) at the beginning of the the next time slot.
There can be some time slipping, and the IEEE Standard provides an uncertainty window of 20 μs. So the packet has a tolerance on its arrival time: it can arrive up to 10 μs earlier and up to 10 μs later the exact receive timing.

When the master sends a packet, there is no frequency hop until the end of the packet's transmission. If the packet lasts only one time slot, we have a frequency hop every time slot; since the time slot duration is 625 μs, the hopping rate will be 1600 hops/s.
If the packet is longer and occupies more than one time slot, the frequency channel used for the next packet (in the next time slot) is the one determined by the CLK value for that slot. So many RF channels can be unused in these cases.

The 1600 hops/s hopping rate is called "nominal rate", and it is the maximum hopping rate. If a packet is longer and lasts more than one time slot, the hopping rate will be inferior.
In fact there is a higher hopping rate of 3200 hops/s, but this hopping rate is used only in the inquiry substate and in the page substate, that means with very small packets used for the connection establishment. In these cases the inquiry/paging

device transmits on two different hop frequencies in a single time slot.

There is also the "same channel mechanism", that is used with the adapted channel hopping sequence. In this case the pseudo-random hopping sequence has a lower number of channels, and the slave responds to the master in the same frequency channel used for the master's packet. So there is no hop until the master sends another packet.

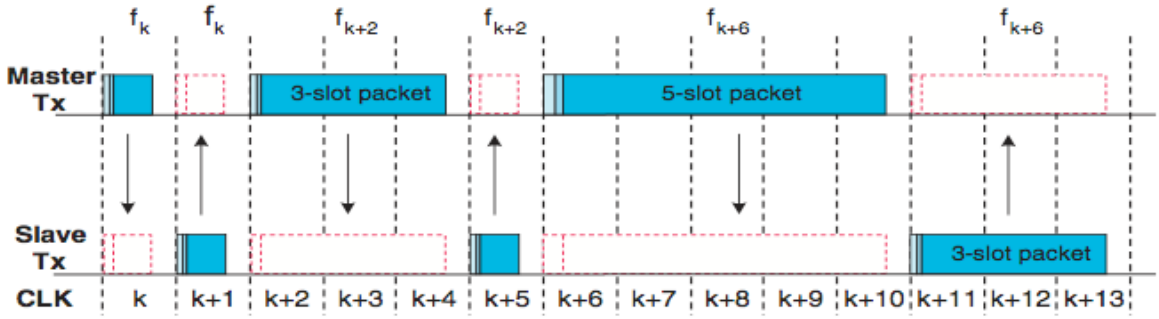

Figure 2.10 – Same channel mechanism

In a multislave configuration the TDD scheme is realized as follows:

- Only the slave that is addressed by one of its Logical Transport Addresses (LT_ADDRs, described later) can return a packet in the next slave-to-master time slot
- If no valid packet header is received, the slave can respond only in its reserved Synchronous Connection-Oriented (SCO, described later) or extended Synchronous Connection-Oriented (eSCO, described later) slave-to-master time slot
- When a broadcast message is received, no slave shall return a response packet

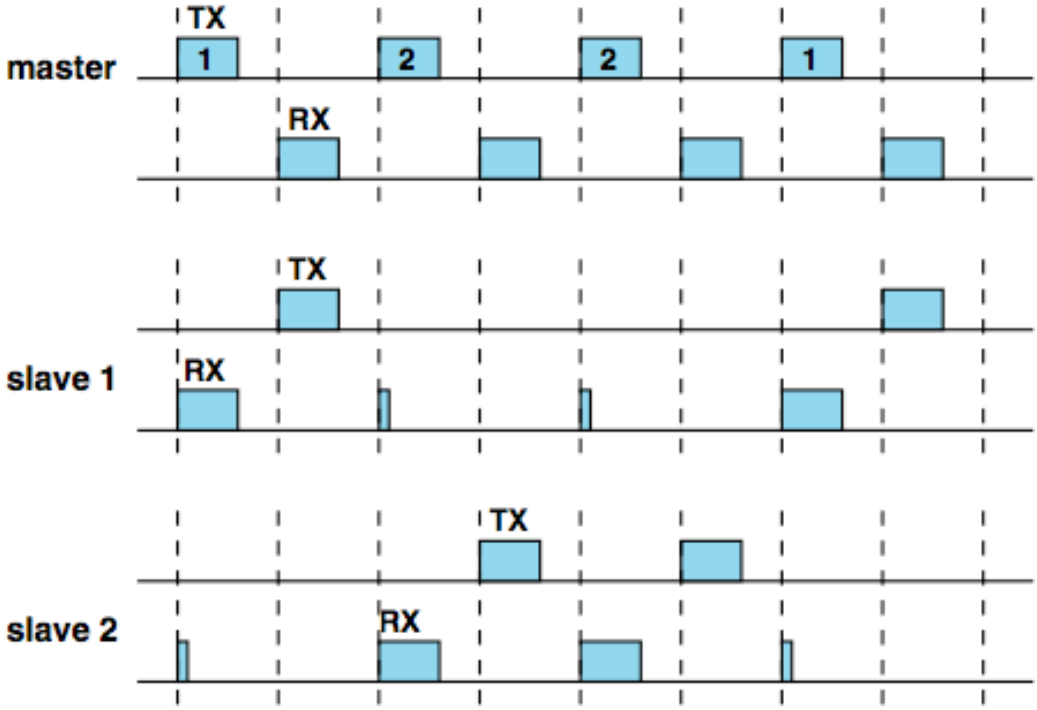

Figure 2.11 – Slot timing in a multislave configuration

2.4: Protocol stack

The data transport system follows a layered architecture.

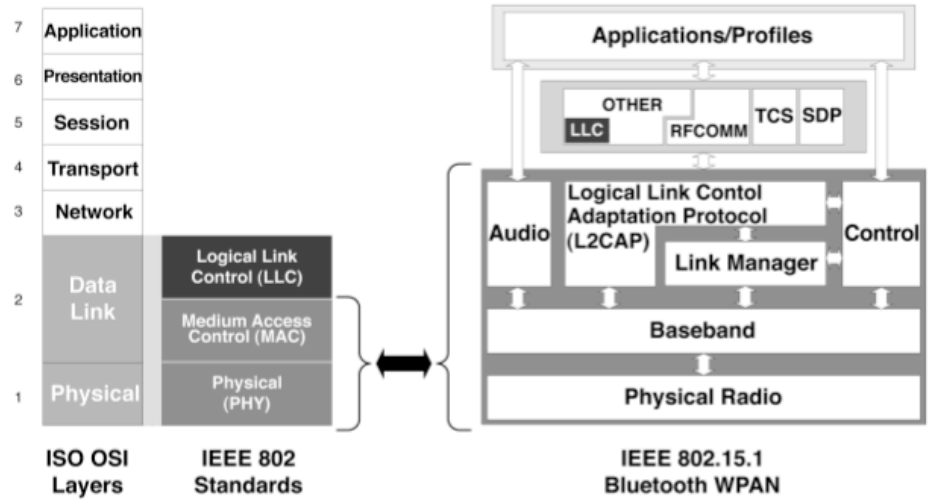

Figure 2.12 – Bluetooth protocol stack and the ISO OSI layers model

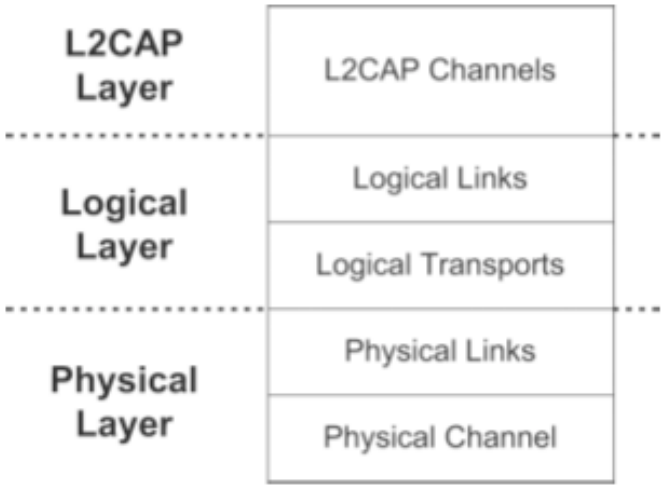

Figure 2.13 – Division of the architectural layers

We have to consider that a Bluetooth device is designed to work in noisy environments, that are therefore unreliable. For this reason there are several levels of protection, added at each architectural layer.

2.4.1: Physical channel

The physical channel is the lowest architectural layer defined in the IEEE Standard 802.15.1 – 2005.

A physical channel is defined by 4 parameters:

1. Frequency hopping sequence
2. Slot timing
3. Access code
4. Packet header

Four physical channels are defined:

1. Inquiry scan physical channel
2. Page scan physical channel

3. Basic piconet physical channel
4. Adapted piconet physical channel

The inquiry scan physical channel is used for discovering devices.
The page scan physical channel is used for connecting devices.
The basic and adapted piconet physical channels are used for communication between connected devices.

A Bluetooth device can use only one physical channel at any given time, and uses Time Division Multiplexing (TDM) between the channels to support multiple concurrent operations.

2.4.2: Physical link

A physical link represents a baseband (BB) connection between devices and it is always associated with exactly one physical channel.

2.4.3: Logical transport

A logical transport is a point-to-point connection between a master and a slave.

The IEEE Standard 802.15.1 – 2005 defines 5 types of logical transport:

1. ACL (Asynchronous Connection-oriented)
2. SCO (Synchronous Connection-Oriented)
3. eSCO (extended Synchronous Connection-Oriented)
4. ASB (Active Slave Broadcast)
5. PSB (Parked Slave Broadcast)

The synchronous logical transports are typically used to transport time-bounded

information, for example voice, or general synchronous data. They can rely on reserved slots at regular intervals, provided by the master. For this reason they can be considered as circuit-switched connections.
In addition the eSCO logical transport may have a retransmission window after ther reserved slots.

Non-reserved slots can be used for asynchronous logical transports, that can be considered as packet-switched connections.

Every active slave in a piconet (but not the master) has another address, related to the logical transport: it is the Logical Transport Address (LT_ADDR). The LT_ADDR is composed by 3 bits, and the all-zero LT_ADDR is reserved for broadcast messages. It is assigned by the master to the slave when it becomes active and it is carried in the packet header.
For every eSCO logical transport a secondary LT_ADDR is assigned to the slave.
A slave accepts a packet only if it matches with its LT_ADDR (primary or secondary), or if it is a broadcast packet.

2.4.4: Logical link

A logical link is associated with a logical transport.

The logical links defined in the IEEE Standard are four:

1. LC (Link Control)
2. ACL-C (ACL Control)
3. ACL-U (ACL User)
4. SCO-S / eSCO-S (SCO/eSCO Stream)

LC and ACL-C are used for control, while ACL-U, SCO-S and eSCO-S are used for user data.
To be more precise, SCO-S and eSCO-S are for synchronous links, and ACL-U is for

asynchronous and isochronous links. (Isochronous links differ from synchronous links because they can also have a non-constant data rate; synchronous links have a constant data rate).
The LC logical link is carried in the packet header, the other logical links are carried in the packet payload.

2.5: The packets

2.5.1: The packets

Data are transmitted in packets. Every packet has a fixed structure, and is formed by 3 parts: the access code, the header and the payload.

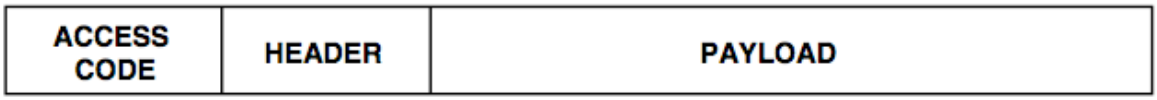

Figure 2.14 – Structure of a general Bluetooth packet

The access code can be of 68 or 72 bits and carries information about the physical channel.
The header is composed by 54 bits; it can be divided into two fields: the packet header and the payload header. The first one carries the logical transport identifier, while the second one carries the logical link identifier.
The payload has a variable size between the minimum value of 0 and the maximum value of 2745 bits; it carries information from higher layers or user data.

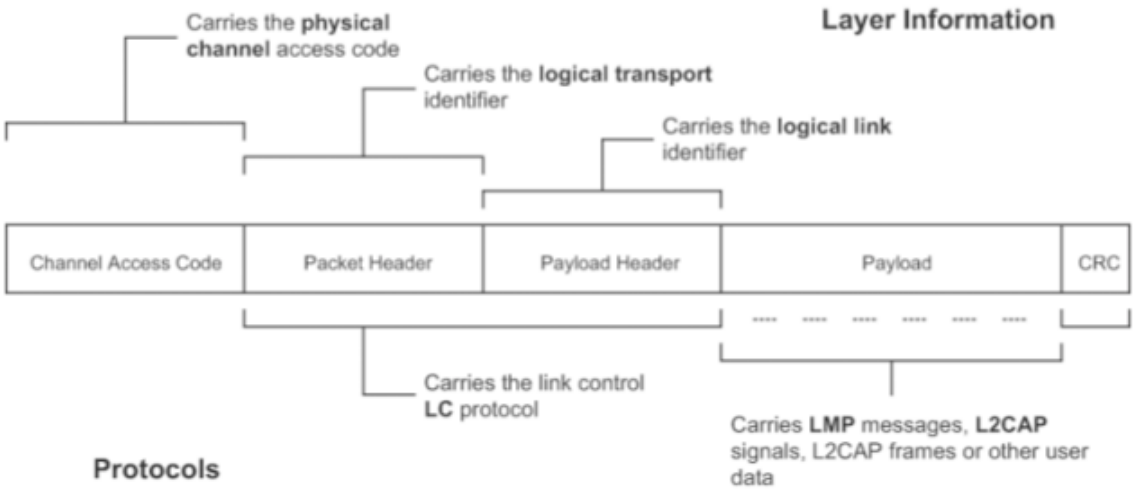

Figure 2.15 – Information carried by the packet's fields

Talking about the packet's bit ordering, the little Endian format is used, so the LSB is the first bit sent over the air and, in illustrations, it is shown on the left side.

According to their length, we can divide the packets in 3 categories:

1. The shortened access code only (68 bits)
2. The access code and the header
3. The full packet (access code, header and payload)

The packets of the first two categories last only one time slot.
The full packet, instead, can last 1, 3 or 5 time slots, according to its length. It is not permitted a packet duration of 2 or 4 time slots to mantain the correct alternation of master's packets and slave's packets. We remember, indeed, that the master can begin a transmission only in even time slots, while the slave can in odd time slots. If an even value of time slots occupation could be possible, this alternation would be damaged.

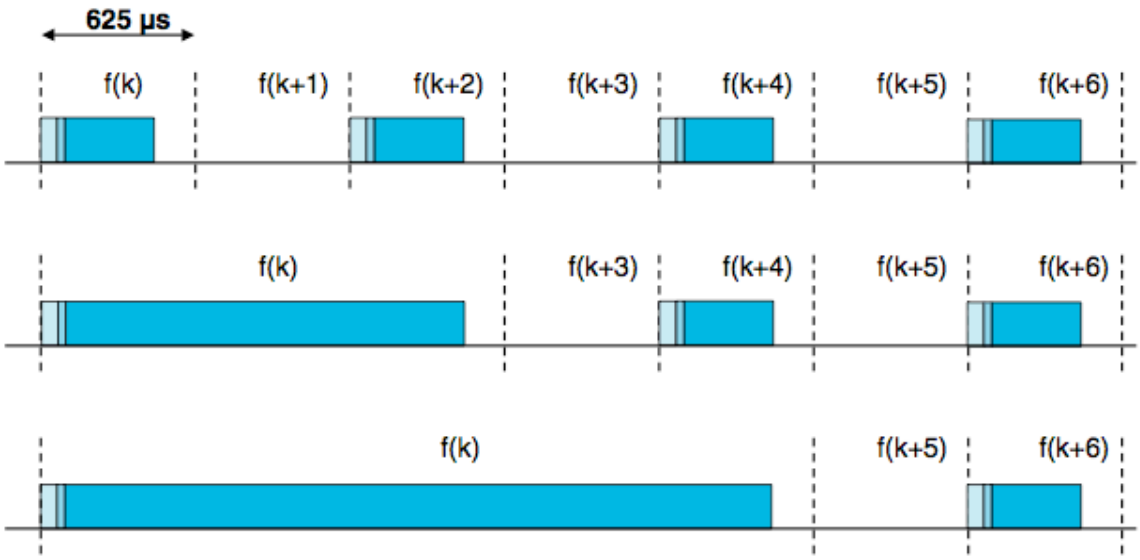

Figure 2.16 – Different packet durations

The IEEE Standard defines 15 types of packet for each of the 3 logical transports: SCO, eSCO, ACL. They are distinguished by the 4-bit TYPE field in the header. The table below shows all these packet types.

Segment	TYPE code $b_3b_2b_1b_0$	Slot occupancy	SCO logical transport	eSCO logical transport	ACL logical transport
1	0000	1	NULL	NULL	NULL
	0001	1	POLL	POLL	POLL
	0010	1	FHS	Reserved	FHS
	0011	1	**DM1**	Reserved	**DM1**
2	0100	1	Undefined	Undefined	**DH1**
	0101	1	**HV1**	Undefined	Undefined
	0110	1	**HV2**	Undefined	Undefined
	0111	1	**HV3**	**EV3**	Undefined
	1000	1	**DV**	Undefined	Undefined
	1001	1	Undefined	Undefined	**AUX1**
3	1010	3	Undefined	Undefined	**DM3**
	1011	3	Undefined	Undefined	**DH3**
	1100	3	Undefined	**EV4**	Undefined
	1101	3	Undefined	**EV5**	Undefined
4	1110	5	Undefined	Undefined	**DM5**
	1111	5	Undefined	Undefined	**DH5**

Table 2.1 – Packet types

The ID packet (not shown in the table) consists of the Device Access Code (DAC) or the Inquiry Access Code (IAC), and has a fixed length of 68 bits.

The NULL packet has no payload: it consists only of the access code and the header. Its length is fixed too: 126 bits. It is often used as acknowledgment, and does not require an acknowledgment.

The POLL packet is similar to the NULL packet: it does not have a payload, but it is used for polling and it does require an acknowledgment.

The FHS (Frequency Hop Synchronization) packet is a control packet that contains, among other things, the device address and the clock of the sender.

The master can also send a packet directed to all the slaves in the piconet: this is

called a broadcast packet.
A broadcast packet shall not be acknowledged, and for this reason the master sends it multiple times, to be (quite) sure it will be received even in a noisy environment. The number of times the packet will be sent, N_{BC}, is a fixed number.

2.5.2: Access code

To mitigate the unwanted effects of a physical channel collision, that may happen, each transmission on a physical channel starts with an access code, that is used as a correlation code by devices tuned to the physical channel. For this reason the access code is always present at the start of every transmitted packet.
It also indicates to the receiver the arrival of a packet, and is used for timing synchronization, offset compensation and identification.
The access code carries information about the physical channel, and all packets sent in the same physical channel are preceded by the same access code.

The IEEE Standard defines three different access codes:

1. IAC (Inquiry Access Code)
2. DAC (Device Access Code)
3. CAC (Channel Access Code)

All of them are derived from the LAP of the BD_ADDR.

The IAC is used in the inquiry substate, and can be distinguished in GIAC (General IAC), for general inquiry, and in DIAC (Dedicated IAC), for dedicated inquiry. There is one GIAC and 63 DIACs, according to the 64 LAP reserved values for inquiry operations
The DAC is used in the page, page scan, master response and slave response substates. The DAC is derived from the paged device's BD_ADDR.
Finally, the CAC is used in the connection state, and is derived from the master's BD_ADDR.

The access code consists of a preamble, a sync word and a trailer (but the latter is not always present).
Both preamble and trailer are fixed zero-one patterns of four bits.

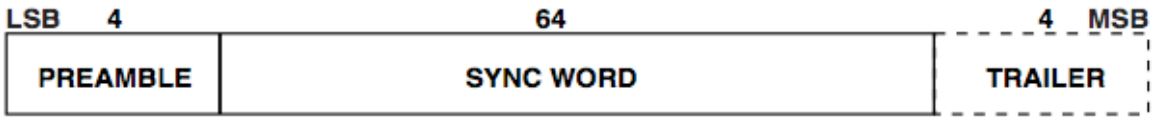

Figure 2.17 – Structure of the access code

As said before, the access code can be of 68 or 72 bits. If a packet header follows, it is 72 bits long, otherwise is 68 bits long.
This short version of the access code is called "shortened access code", and does not contain a trailer. It is used in inquiry substate, page substate and park state.

To generate the sync word of the access code the IEEE Standard specifies the use of a (64, 30) expurgated block code and of a pseudo-random noise (PN) sequence.
The code guarantees large Hamming distance ($d_{MIN} = 14$) between the sync words based on different addresses, while the PN sequence improves the auto-correlation properties of the access code.

This block code is generated starting from a BCH (Bose-Chaudhuri-Hocquenghem) code.
The BCH codes are cyclic codes, which means block codes with a large amount of mathematical structure. Thanks to this mathematical structure, easy encoding operations and simple decoding algorithms are possible [10].

In particular, for any pair of positive integers *m* and *t* there is a BCH code with these parameters:

$$n = 2^m - 1 \qquad n - k \leq mt \qquad d_{MIN} \leq 2t + 1$$

where k is the length (in bits) of the word to encode, and n is the length (in bits) of the encoded word.
This code can correct *t* or fewer errors.

The BCH code is one of the most useful for correcting random errors because decoding algorithms can be implemented with an acceptable amount of complexity. Therefore it is widely used.

The expurgated block code used in the Bluetooth access code generation has a generator polynomial that comes out from the (63, 30) BCH generator polynomial "157464165547" (expressed in octal notation).
It has been modified to obtain a (64, 30) code: this means the length of the input word is 30 bits, and the block returns a 64-bits-long word.

2.5.3: Header

The header is formed by 18 bits and is encoded with a 1/3 rate FEC, resulting in a 54-bit header, as said before.

It can be divided in 6 fields:

1. LT_ADDR (Logical Transport Address) 3 bits
2. TYPE (type code) 4 bits
3. FLOW (flow control) 1 bit
4. ARQN (acknowledge indication) 1 bit
5. SEQN (sequence number) 1 bit
6. HEC (Header Error Check) 8 bits

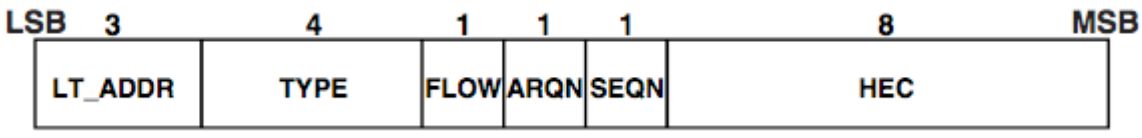

Figure 2.18 – Structure of the header

2.5.4: Bitstream processing

Before the packet is transmitted over the air, it has to be processed to increase its reliability and security. This is done through a process called "bitstream processing".

The bitstream processing is not applied to the access code, and it is different for the header and for the payload.

For the header, a field of Header Error Check (HEC) is added, then there is a scrambling operation with a whitening word, and at the end there is Forward Error Correction (FEC) encoding. Naturally at the receiver side the same operations, but inverted and in the inverse order, are done to the received packet header.
All these processes are mandatory.

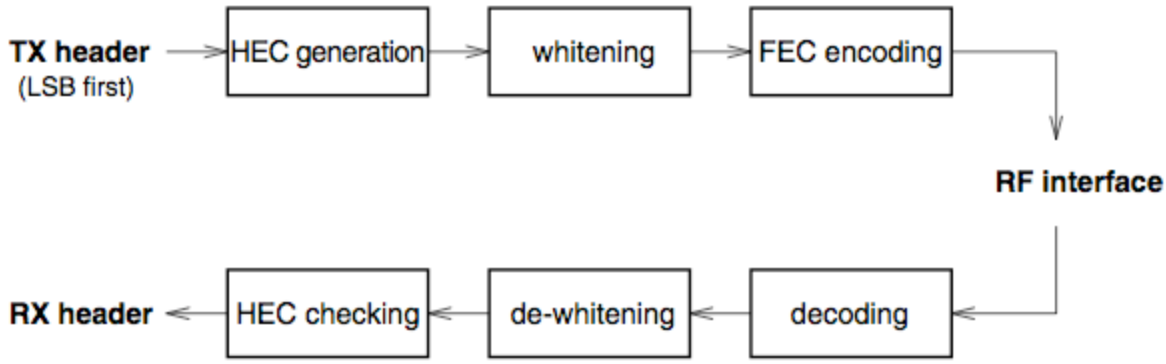

Figure 2.19 – Header bitstream processing

For the payload, instead, only the scrambling with a whitening word is mandatory (and, of course, de-whitening at the receiver side). There are other processes that can be optionally done: a Cyclic Redundancy Check (CRC), an encryption and an encoding for the error correction or detection. As usual, if these operations are applied to the payload, the receiver has to do the inverse operations when it receives the packet.

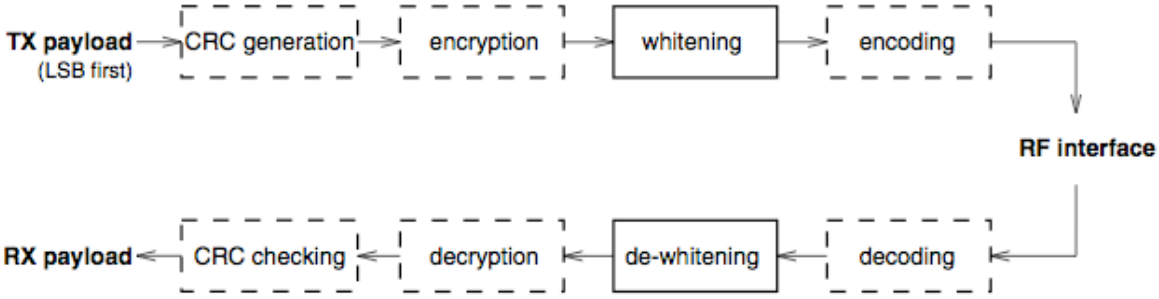

Figure 2.20 – Payload bitstream processing

The error corrections schemes defined by the IEEE Standard 802.15.1 - 2005 are three:

1. 1/3 rate FEC
2. 2/3 rate FEC
3. ARQ (Automatic Repeat Request)

While the other operations have the scope to protect the data from errors and from unauthorized receivers (the encryption), the scrambling with the whitening word is done to randomize the data from highly redundant patterns and to minimize DC bias in the packet.

2.6: States and substates

For every Bluetooth connection the IEEE Standard 802.15.1 - 2005 describes three major states and seven substates, in which a device can be.

The three states are:

1. Standby
2. Connection
3. Park

The seven substates are:

1. Inquiry
2. Inquiry scan
3. Inquiry response
4. Page
5. Page scan
6. Master response
7. Slave response

The substates are interim states, that is to say states in which the device remains for a limited period of time, and are used for other devices discovery and for connection establishment.

States and substates refer to a single connection, so a device can be simultaneously in many states (or substates), one for each connection it has.

This state diagram shows an overview of states and substates:

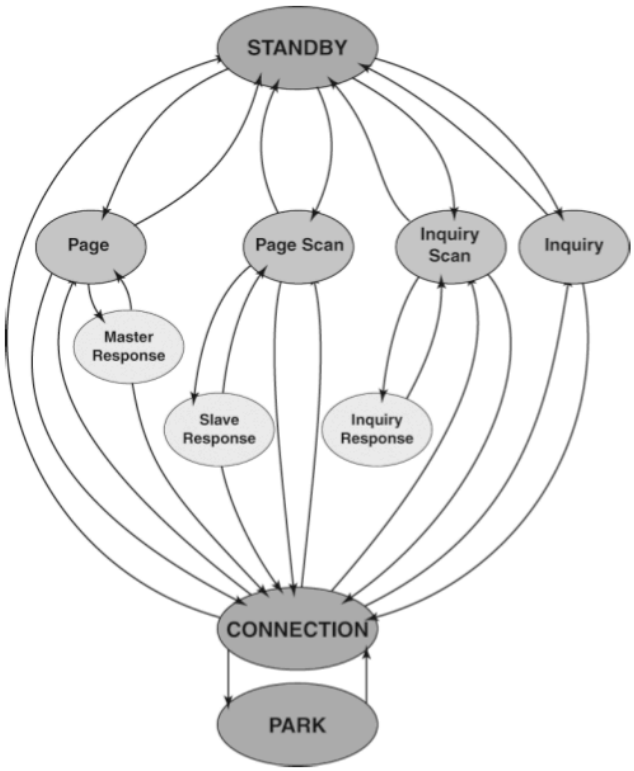

Figure 2.21 – States and substates

It should be noted that a device cannot move from a state straight to another one it chooses. There are some constrained passages from a state to another it should respect, and these passages reflect the logical changes in a connection establishment and, later, in the connection management.

Here is a typical progress through the states the master and a slave follow for a single connection:

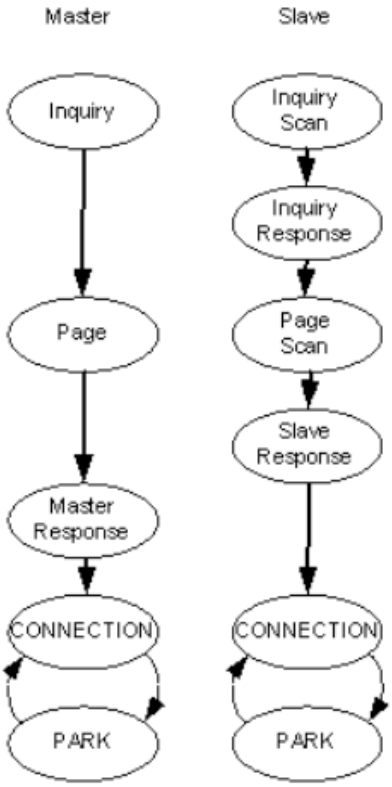

Figure 2.22 – Typical progress through the states

The **STANDBY** state is the default state in which a Bluetooth device can be. During the permanence in this state it is in a low-power mode. When the device wants to connect to other devices, it leaves the standby state and enters the inquiry or inquiry scan subtstate.

In the **CONNECTION** state, a device has already established a connection with another device: they are physically connected to each other within a piconet and they

can send and receive data packets.
Three different modes are defined for the connection state:

1. Active mode
2. Sniff mode
3. Hold mode

In the ACTIVE mode, that is the main mode of the connection state, both master and slave actively participate in the communication and exchange of data packets. The slaves are also called active slaves.
If a device is not going to be nominally present on the channel at all times, it may enter the sniff mode or the hold mode.
In the SNIFF mode the slave device listens with less frequency for incoming packets from the master. It defines a duty cycle of periods of presence and absence, entering a reduced power mode.
When a slave device is in the HOLD mode, it does not (temporarily) support an ACL connection, but only synchronous links.

A slave enters the **PARK** state when it does not actively participate in the communication. As the state's name says, the device is parked, not active, and thanks to this it can save energy entering a low-power mode. Even if not active, it is still connected to the master: it periodically reactivates to mantain the synchronization and to control if there are any broadcast packets.
The park state has also another use: only up to seven slaves can be active in a piconet, but more slaves can remain connected in park state. These parked slaves are not active on the channel, but are synchronized to the master and can return active again without the connection establishment procedure. Naturally only 7 slaves can be active in a certain instant, and the other must be parked.

In order to mantain the parked slaves synchronized to its clock, the master periodically sends a signal called beacon train. This signal is periodic and is used by the parked slaves not only to stay synchronized, but also for broadcast messages to parked slaves and to exit the park state (unpark).
The beacon train is composed by N_B beacon slots (with $N_B \geq 1$) which are temporally

equidistant. They are tranmsitted periodically with a constant temporal interval T_B. The values of N_B and T_B are not specified by the IEEE Standard, but are chosen in order to have sufficient beacon slots so that a parked slave can synchronize its clock in a certain temporal window, even in an error-prone environment.

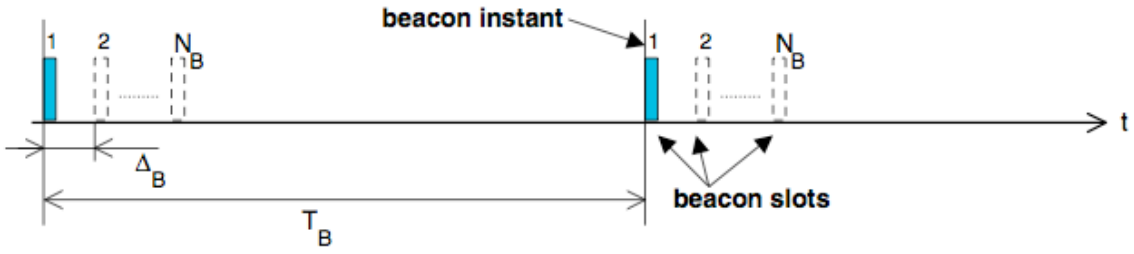

Figure 2.23 – Beacon train

Parked slaves can also use only a subset of beacon slots, with a periodicity multiple of T_B. They can do that to save even more energy, beacuse in this case they listen to the channel with even less frequency.

The **INQUIRY**, **INQUIRY SCAN** and **INQUIRY RESPONSE** are substates used to discover other devices present in the area and to establish a first kind of connection with them. After this procedure, the inquiry device (which later will become the master) knows the clocks and the addresses of the inquiry scanning devices (the future slaves) that answered, that is to say that entered the inquiry response substate. The inquiry procedure is asymmetrical: the inquiry device carries out the discovery of other devices, while the inquiry scanning device must be discoverable. If it wants to accept a connection, it enters the inquiry response substate and answers to the (future) master.

The **PAGE**, **PAGE SCAN**, **MASTER RESPONSE** and **SLAVE RESPONSE** substates are used to establish a connection. In order to do that, the paging device (the future master) needs the address of the page scanning device (the future slave), obtained from the inquiry procedure or from a previous connection with this device.
As the inquiry procedure, the paging procedure is asymmetrical: the paging device carries out the connection (page) procedure, while the page scanning device must be

connectable. Then the slave answers to the master (slave response), who answers back again to the slave (master response). After these passages, both the devices enter the connection state.

There is also a role switch procedure, thats is used, as the name says, to swap the roles of two devices connected in a piconet: the master becomes slave and the slave becomes master. This implies moving from the physical channel defined by the original master to the one defined by the new master.
After the role switch the original piconet physical channel may cease to exist or may be continued, if the original master had other slaves that are still connected to the channel. In the latter case, a scatternet is created, formed by the old and the new piconets.

2.7: Temporal regularity: the inquiry substate

In order to extract characterizing temporal features in the Bluetooth packets, we have to analyze the temporal regularities present in this technology.
As can be seen in the IEEE Standard 802.15.1 – 2005, there are very few temporal regularities, and, in addition, most of them depend on parameters that are fixed but not specified by the Standard. This means we do not know them.

An example of this is the beacon train, with its N_B and T_B unspecified values.

Another example is the slot timing and the packet duration. A packet can last 1, 3 or 5 time slots, according to its length, which is not known. To be more precise: a packet can occupy 1, 3 or 5 time slots, but it lasts a time t_0, according to its length.
If $t_0 \leq T_S$ it occupies 1 time slot.
If $T_S < t_0 \leq 3\ T_S$ it occupies 3 time slot.
If $3\ T_S < t_0 \leq 5\ T_S$ it occupies 5 time slot.
(If the packet is longer, it should be divided in two or more parts, so it can respect this specification of the Standard).

We can not see a strong timing regularity in this mechanism.

We can find temporal regularity in the inquiry substate. This is quite normal because it is the initial phase a Bluetooth device uses to discover the other devices eventually present in the area and to start, later, a connection. Before this first discovery, the device shall not know anything about the other devices, and for this reason the procedure presents a temporal regularity any device can detect.

When a device enters the inquiry substate, it transmits repeatedly and in broadcast the ID packet at different hop frequencies. This hopping sequence (the inquiry hopping sequence) covers 32 of the 79 possible frequency channels, and is derived from one of the reserved LAPs (it depends if this inquiry is general or dedicated).

The ID packet, in this case, is the IAC of 68 bits. In the IEEE Standard 802.15.1 – 2005, from section 8.6.3.2 to section 8.6.3.3.2, it is exactly described how to generate it.
Because of the bitrate of 1 Mb/s, the ID packet duration is 68 μs, and it is repeated twice in a time slot. The following time slot is dedicated to receive eventual responses from devices which are in the inquiry scan substate and want to answer, entering the inquiry response substate.
So we have a time slot alternation: one is for inquiry transmission and one is for scan for inquiry response messages.

The inquiry timing is shown in the figure below, where we can visually appreciate the temporal regularity:

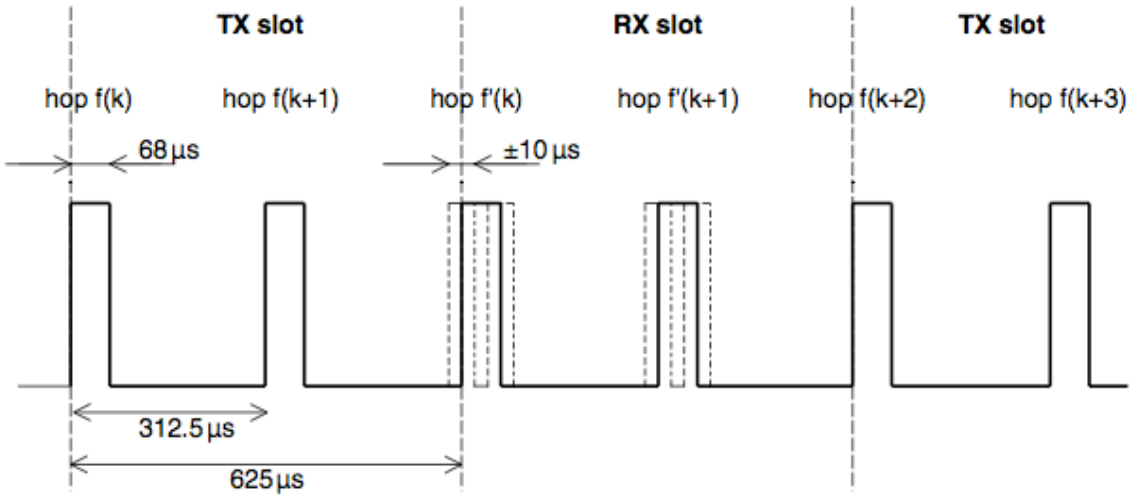

Figure 2.24 – Inquiry timing

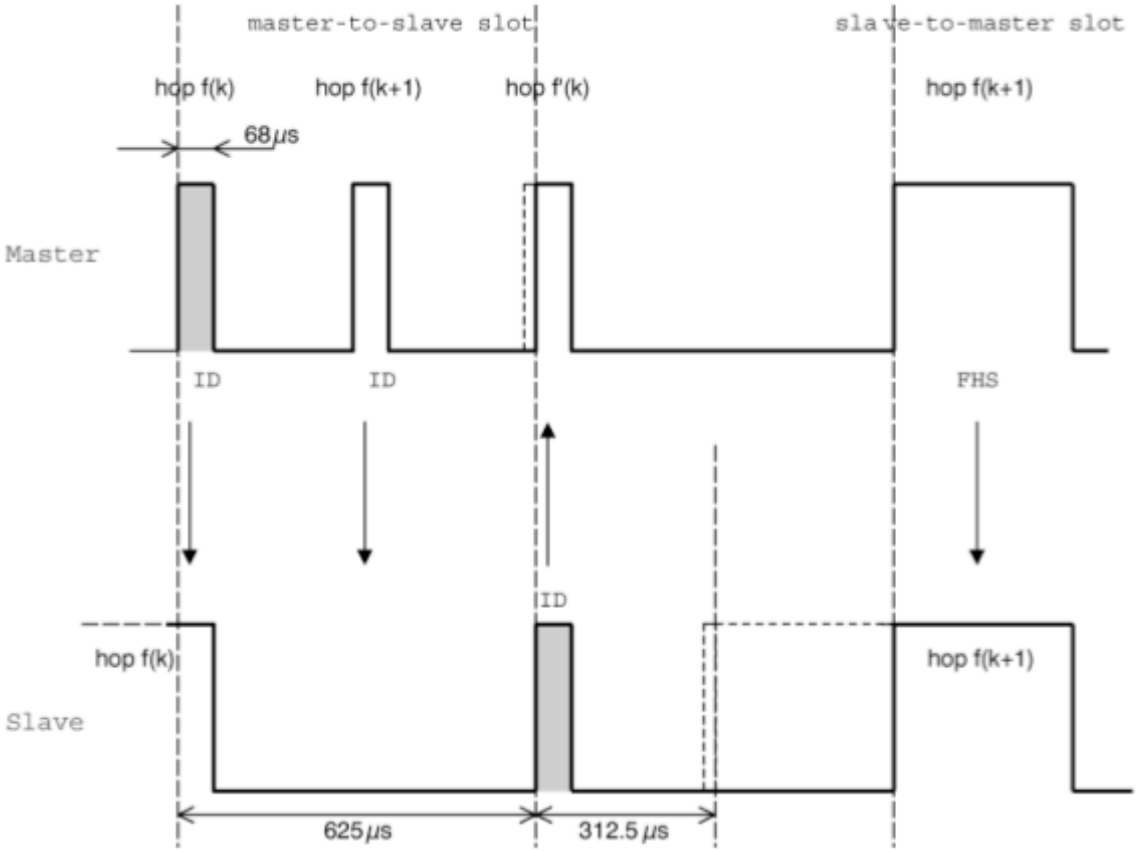

Figure 2.25 – Timing of inquiry response packets on successful inquiry in first half slot

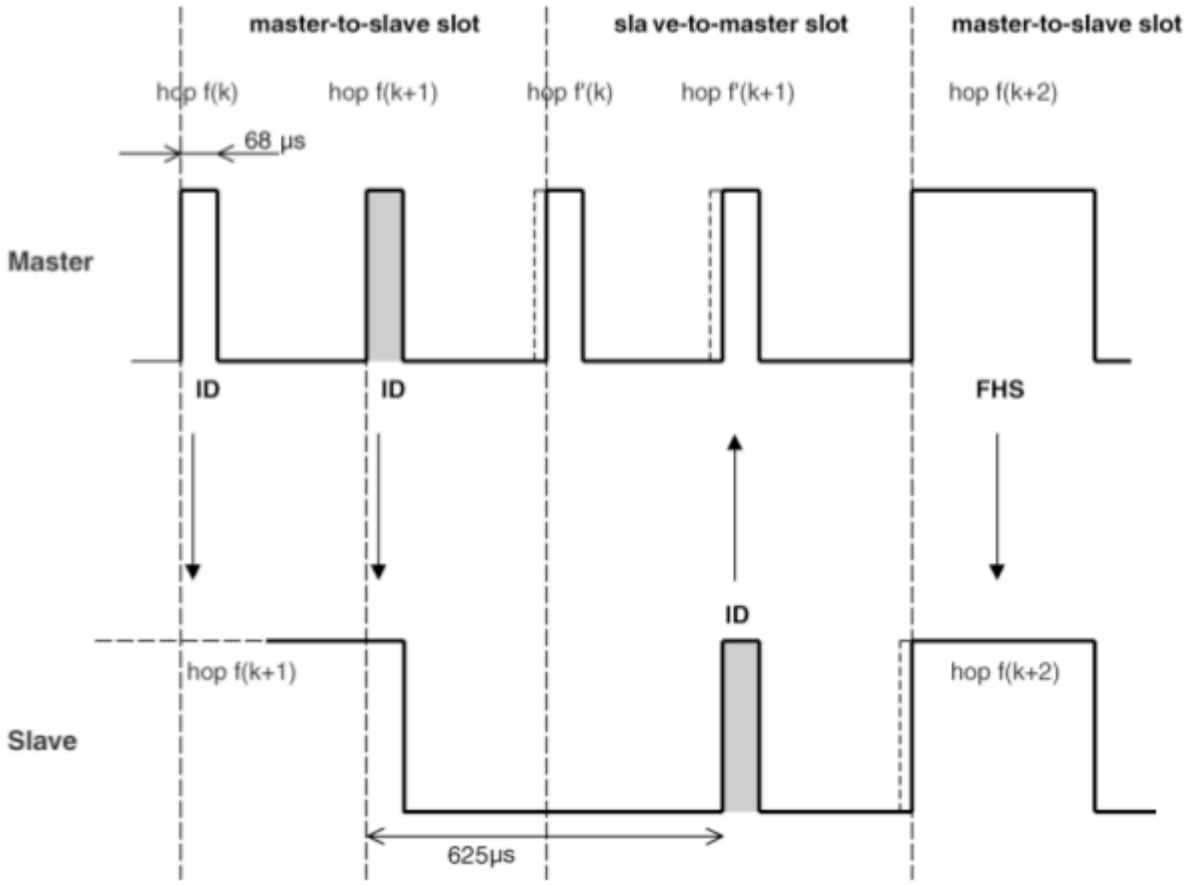

Figure 2.26 – Timing of inquiry response packets on successful inquiry in second half slot

The packet used for the inquiry response is the FHS packet. Thanks to this packet, the inquiry device collects addresses and clocks of the inquiry response devices.

A device remains in the inquiry substate until the “inquiryTO” timer expires, unless the BB resource manager decides the inquirer has collected a sufficient number of responses. The “inquiryTO” timer defines the number of slots the inquiry substate will last. Its value is not fixed, but the IEEE Standard recommends a value of 10.24 s. The resource manager can also decide the inquiry substate can last more time, to collect more responses. This is done when the environment is very noisy.

It is not specified when a device should enter the inquiry substate, but if it does this periodically, the IEEE Standard recommends a period of about one minute.

CHAPTER 3
SIMULATING BLUETOOTH TRAFFIC

3.1: Using MATLAB

After seeing a brief introduction to the IEEE Standard 802.15.1 – 2005, the main characteristics of the Bluetooth technology are known.
Now a simulation of the behaviour of a Bluetooth transmitter can be done, because its functional operations are known, and packets and the relative signals can be generated.
After finding some characterizing features, that can be useful for our purpose, the Bluetooth packets and the signals can be generated; an analysis of them can be done, to recognize the features.
In other words the scope is to extract features that will let us detect, with a reasonable certainty, if the traffic a generic device is capturing is Bluetooth traffic or not.

To simulate Bluetooth traffic in this work, MATLAB and Simulink have been used.

MATLAB stands for "Matrix Laboratory" and is a numerical computing environment and high-level programming language.
Developed by “The MathWorks”, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, and Fortran.
Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

This software application has been chosen because of its ductility and reliability, and because it is widely used across industry and the academic world. MATLAB users come from various backgrounds of engineering, science, and economics.

3.2: The Bluetooth transmitter

The first step is, therefore, to create a Bluetooth transmitter in MATLAB.

Simulink, an additional package included in MATLAB, contains a demo of a Bluetooth transceiver. Its block diagram is shown below:

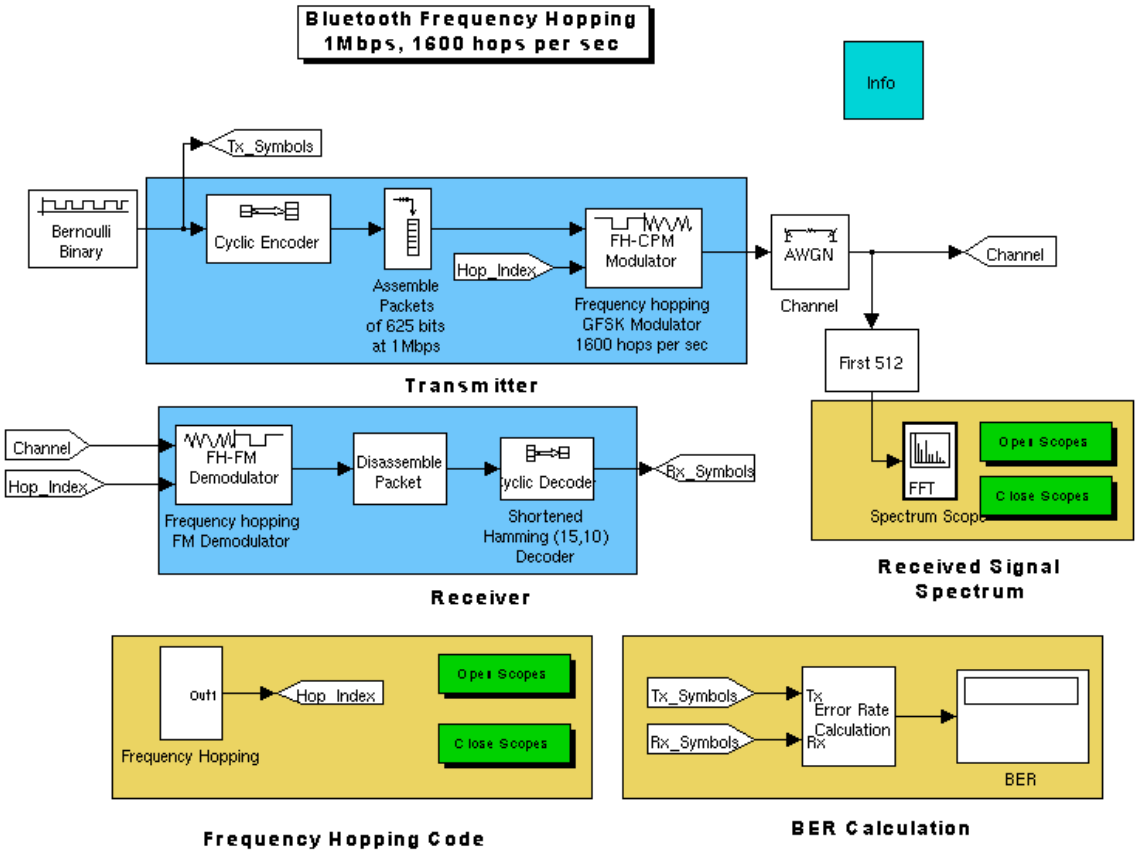

Figure 3.1 – Block diagram of Simulink's Bluetooth transceiver

This demo simulates an entire Bluetooth transmission, that is to say a transmitter, an Additive White Gaussian Noise (AWGN) channel and a receiver. The transmitter and, of course, the receiver incorporate the frequency hopping system, with a hopping rate of 1600 hops/s. The hopping sequence is not derivated from the address and clock of one of these two simulated devices (supposing it is the master); the frequencies are in fact chosen randomly.
The bitrate this model simulates is of 1 Mb/s. The bits are randomly chosen too.
This model also contains a Bit Error Rate (BER) calculation block and shows the signal progress in the frequency domain, calculating the Fast Fourier Transform (FFT) of the signal.

This entire model is not necessary for our scope.
What is needed for the work is only the transmitter, but that is able to send chosen bits.

Considering the cognitive radio, the device a priori does not know the frequency hopping sequence, and therefore does not know to which 1 MHz channel in the ISM entire band to be tuned to. What it can do, is to consider the entire 2.4 GHz ISM band; to be more precise, the 79 MHz band formed by the 79 Bluetooth frequency hopping channels, i.e. from 2401.5 MHz to 2480.5 MHz.
In this way it can be considered two different conditions, that are equivalent for the scope of this work.
The first one is that the generic device that has to capture the signals and, later, classify the captured traffic, knows somehow to which frequency to be tuned to in every instant; the second one is that it is able to capture in all the entire 2.4 GHz ISM band.
This means that it knows the hopping sequence (first condition) or has 79 filters, each tuned to a Bluetooth frequency channel (second condition).

For this reason the frequency hopping results “transparent” for our scope. It is transparent because it is like the device does not see there is a frequency hopping, it only receives the signal as it would be always in the same frequency.

On the contrary, an important aspect is that it is necessary for this work to choose exactly which bits will be transmitted. This is needed because we have to find characterizing features, and these features may depend on which bits the packets carry.

Therefore a new model of the Buetooth transmitter has been created in MATLAB, with the characteristics useful for this work.
In this Bluetooth transmitter model, chosen bits are modulated with an FSK modulation (equivalent, for our scope, to the GFSK modulation expected by the Standard), then they are modulated in the ISM band and finally the resulting signal is sent over the air.

3.3: Which features

As said above, it is important to choose exactly which bits will be transmitted, because in the bit sequeneces some features can be found.

In fact it must be always present that the recognition of the Bluetooth signals is here tempted through feature detection.
Therefore the characterizing features must be chosen, that is to say something that can lead to the recognition of the Bluetooth technology.

First of all, the modulation used in the Bluetooth technology is the GFSK modulation; therefore the output signal has a constant envelope. This can be a first degree of discrimination (even if it is not strong enough to call it “feature”): all signals with non-constant envelope can not be Bluetooth signals.

Another important thing are the known and fixed bit sequences that the IEEE Standard 802.15.1 establishes. This can be an important feature, that can distinguish

Bluetooth from other technologies, and therefore let us recognize it in the environment.
Our Bluetooth transmitter model must be able to recreate these sequences, and then a classifier must be able to recognize them, to recognize this technology among the others. In other words, the classifier must use these features to recognize Bluetooth signals.

3.4: Creating the packets and the signals

The first step is to create Bluetooth packets.
The packets used in this simulation are the ID packet and the access code (even if it is not properly a packet, but a bit sequence that is present at the start of every packet).

The ID packet used in the inquiry substate is the only packet that has fixed and a priori-known bits. As described in chapter two, this packet consists of the Inquiry Access Code (IAC), and has a fixed length of 68 bits.
The standard defines every single pass to create a generic access code. To generate it, the Lower Address Part (LAP) of the device is needed. In this case, for the General Inquiry Access Code, there is a reserved LAP, whose value is 0x9E8B33 (expressed in hexadecimal notation).
Then this initial bit sequence is properly manipulated to obtain a final sequence of 68 bits.
As seen in more detail in the second chapter, there are two main operations made to generate the sync word of the access code: the IEEE Standard specifies the use of a (64, 30) expurgated block code and of a pseudo-random noise (PN) sequence.
The code guarantees large Hamming distance ($d_{MIN} = 14$) between the sync words based on different addresses, while the PN sequence improves the auto-correlation properties of the access code.
The expurgated block code used in the Bluetooth access code generation has a generator polynomial that comes out from the (63, 30) BCH generator polynomial

“157464165547” (expressed in octal notation). It has been modified to obtain a (64, 30) code: this means the length of the input word is 30 bits, and the block returns a 64-bit-long word.

At the end, this 68-bit sequence is obtained.
This ID packet is important for two reasons:

1. We have a known fixed sequence that can identify Bluetooth
2. It is transmitted repeatedly

Therefore, the ID packet can be a good characterizing feature for Bluetooth.

The access code is composed by 72 bits, and is present at the start of every Bluetooth packet. Its generation follows the same passes discussed in the second chapter and above, for the ID packet. That is obvious, because the ID packet is a particular case of a general access code.

The starting bit sequence for the generation of the access code is the LAP of the BD_ADDR of the Bluetooth device, i.e. it depends from the device’s address and is different for every device.
Nevertheless some of these 72 bits are a priori known.
The first 5 bits, in fact, are either “01010” or “10101”. This is because the access code starts with a 4-bit preamble, that is composed by an alternation of ones and zeros, depending from the next bit (the fifth bit, i.e. the first bit of the sync word).
Even the last 11 bits of the access code are known. The last 4 bits are the trailer, that is a fixed zero-one pattern (as the preamble is: “0101” or “1010”). The previous 7 bits are also fixed, because they form a length-seven Barker sequence that is present at the end of the sync word.

For this reason there four possible patterns present in the access code: two possible preambles, and for each two possible 11 final bits.
These known bits, moreover, are at a fixed temporal distance. As the access code is composed by 72 bits, the first 5 bits are separated from the last 11 bits by 56 unknown bits (they depend from the device’s address), i.e. 56 μs.

To test the recogniton step, random bit sequences of 68 and 72 bits are also created.

All these bit sequences, both the known ones and the random ones, are then modulated with an FSK modulation (equivalent, for our scope, to the GFSK modulation expected by the Standard, as explained later).
They are then shifted in band, using every time a random carrier frequency f_c. These possible random f_c are 79, one for each Bluetooth channel. In this way our simulation does not depend on the specific frequncy channel, and it is thought as the generic device does not know the frequency hopping sequence.
To make the simulation more realistic, an Additive White Gaussian Noise (AWGN) channel is taken into consideration.

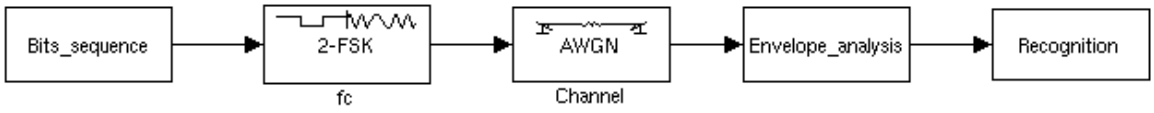

Figure 3.2 – Steps of the simulation

Now the packets and the relative signals are generated, the recognition of Bluetooth signals based on feature detection can be done.

3.5: Recognition of Bluetooth signals

The recognition of Bluetooth signals is based on feature detection. The features are the ones explained above:

- Constant envelope
- ID packet (known 68-bit sequence)
- Access code (known 5+11 bits in a 72-bit sequence)

The recognition is made of these steps:

1. Non constant envelope signals are discarded

2. If the signal's envelope is constant, its bit sequence is estimated through the signal's zero-crossing
3. The estimated bit sequence is compared with the Bluetooth known sequences, and if they are the same sequences, it can be said that the received signal is a Bluetooth signal

It must be observed that the estimation of the bit sequence is done with the zero-crossing technique. For this reason in this simulation the GFSK modulation, used in the Bluetooth technology, is equivalent to the FSK modulation, they have no different effects in the zero-crossing of the signal.
In this simulation is used, therefore, the FSK modulation instead of GFSK. This is done because it permits a simpler implementation.

The estimation of the bit sequence is done with the zero-crossing. To be more precise, the number of zero crossings in a bit interval T_B = 1 μs is calculated, and it is compared to the number of zero crossings in the previous T_B. If they are (quite) the same, the estimated bit is the same bit as the previous bit, otherwise the estimation provides a change of bit.
In this way the estimated bit sequence depends on the intial bit. Therefore, for every received signal, two sequences are estimated, one starting with a "0" and the other starting with a "1", and they are compared with the Bluetooth sequences.

As the AWGN is taken into account, there can be different zero-crossings for the same bit. For this reason the recognition procedure calculates if they are "quite" the same.
A study on the threshold the number of zero crossings must have, to consider a change of bit, is done, and the optimum threshold is found using a linear classifier: the Pocket classifier. In this way the optimum threshold is found for the different values of SNR considered, and it is then used in the sequence estimation.

Moreover, in this work the noise is taken into account only considering different thresholds for different SNR values. Even if not implemented in these simulations, the recogniton step could also be more tolerant, especially with low SNR values and with

the ID packet of 68 bits, considering a bit sequence an ID packet even if some bits are not the same.

There is an hypothesis that has been done in this simulation: a perfect synchronization on the packet's arrival time. In fact the estimation of the bit sequence begins exactly with the initial instant of the signal that represents the first bit of the sequence.
Naturally the generic device that has to do the acquisition and recognition can not have this synchronization, but must probably use a sliding window of acquisition and recognition.

It must be also remembered the ID packet is sent regularly to discover other Bluetooth devices. So, if an ID packet is found, it can be said that there is a Bluetooth device in the environment, but not more. From the ID packet we can not say if a Bluetooth connection is actually active.
If an access code is found, instead, we can say more: in fact the access code is present at the start of every Bluetooth packet, and this means that Bluetooth packets are sent over the air. For this reasons, if an access code is found, it can be said that there is an active Bluetooth connection in the listened environment.

3.6: Higher layers features

Until now the features used in this work are quite of low layers in the protocol stack. In fact physical characteristics of the signal are taken into account, together with sequences of bit.

Higher layers features, for example characteristics of the MAC layer, can lead to simpler algorithms of recognition and could be useful in a generic device (an energy detector, for example), that can be integrated in the cognitive radio.

Unfortunately the Bluetooth technology does not show strong regularities in MAC layer, i.e. strong enough to use them as characterizing features that can lead to recognition.
As described in the second chapter, the most regular behaviour the Bluetooth shows is in the inquiry substate.

Despite this, the temporal behaviour of a Bluetooth packet exchange can be useful, because it presents differences from another large used technology operating in the 2.4 GHz ISM band: the Wi-Fi (IEEE Standard 802.11).

For this reason a simulation of Bluetooth packet exchange has been made. The packets have been generated using MATLAB.

It has been considered a piconet with only two devices, a master and a slave, in connection state.
The two devices send their packets alternately: one device (the master, for example) sends its data packets, and for every received packet the other device (the slave) sends back an acknowledgement. The data packets sent by the master can occupy 1 or 3 or 5 time slots, according to their length, where the time slot lasts 625 μs. On the other hand, the acknowledgement packets are short and occupy only 1 time slot.

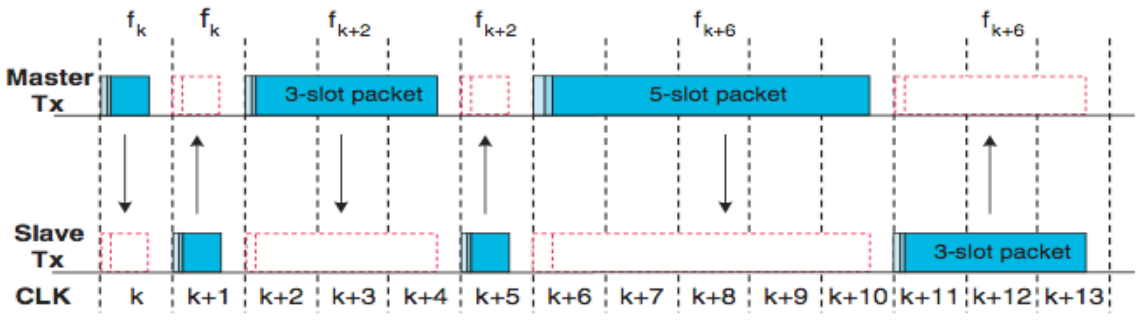

Figure 3.3 - Exchange of different-length packets

The maximum length of a 1-time-slot-packet has been fixed at 366 bits, that correspond to 366 μs.

The maximum length of a 3-time-slots-packet has been fixed at 1622 bits, that correspond to 1622 μs.
The maximum length of a 5-time-slots-packet has been fixed at 2870 bits, that correspond to 2870 μs.
For the acknowledgement a NULL packet has been used. The NULL packet has a fixed length of 126 bits, so its duration is always 126 μs.

Three different scenarios has been considered:

Scenario 1: all the data packets occupy only 1 time slot

Scenario 2: 90 % of data packets occupy 1 time slot
7 % of data packets occupy 3 time slots
3 % of data packets occupy 5 time slots

Scenario 3: 80 % of data packets occupy 1 time slot
15 % of data packets occupy 3 time slots
5 % of data packets occupy 5 time slots

In every scenario the 70 % of the data packets duration is the maximum considered (366 μs for 1-time-slot-packets, 1622 μs for 3-time-slot-packets, 2870 μs for 5-time-slot-packets).
The duration of the other 30 % of packets is uniformely distributed between the minimum value considered and the maximum value considered. For this reason:
126 μs ≤ 1-time-slot-packet duration ≤ 366 μs
1250 μs ≤ 3-time-slots-packet duration ≤ 1622 μs
2500 μs ≤ 5-time-slots-packet duration ≤ 2870 μs

As the standard expects, for every packet's arrival time a jitter of ± 10 μs has been taken, to consider a not perfect synchronization between the two devices. This jitter has a gaussian distribution with mean 0 and standard deviation 10/3 μs. So in this context the 99 % of jitter values are included in the ± 10 μs interval, and the other 1 % of values has been brought back to the extreme jitter values. For this reason in every case the minimum jitter is - 10 μs and the maximum jitter is 10 μs.

Here is a summary of the values used in the three different scenarios considered:

duration of one time slot: 625 μs
duration of the NULL packet: 126 μs
minimum duration of a 1-time-slot-packet: 126 μs
minimum duration of a 3-time-slots-packet: 1250 μs (625*2)
minimum duration of a 5-time-slots-packet: 2500 μs (625*4)
maximum duration of a 1-time-slot-packet: 366 μs
maximum duration of a 3-time-slots-packet: 1622 μs
maximum duration of a 5-time-slots-packet: 2870 μs
jitter: ± 10 μs

This simulation has been used for the Wi-Fi vs. Bluetooth classification [17] [18]. In these works it can be seen that this Bluetooth behaviour, even if it has no strong regularities, can be used to distinguish Bluetooth technology from Wi-Fi technology.

This simulation has been used in the work "Automatic network recognition by feature extraction: a case study in the ISM band". This work has been accepted for publication in *Proceedings of the 5th International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, Special Session on Cognitive Radio and Networking for Cooperative Coexistence of Heterogeneous Wireless Networks, June 9-11 2010, Cannes, France.
The manuscript is inserted in the Appendix.

CHAPTER 4
RESULTS

In this chapter the results of the simulations are presented and commented.

Before doing this, the simulation steps are briefly summarized.

- The bit sequence of the ID packet is generated; all the 68 bits are known
- Four different bit sequences of 72 bits of the access code are generated; the known bits are the first 5 and the last 11. The possible sequences are four because there are two different initial 5-bit patterns and two different final 11-bit patterns
- Random sequences both of 68 and 72 bits are generated
- All these sequences are modulated FSK and shifted in band, with 79 random carrier frequencies f_c in the 2.4 GHz ISM band
- AWGN is added, considering five different conditions of Signal-to-Noise Ratio (SNR): SNR→∞ (no noise), SNR = 40 dB, SNR = 20 dB, SNR = 10 dB, SNR = 5 dB

After these signal generation steps, there is the Bluetooth recogniton step:

- Signal with non-constant envelope are discarded
- The number of zero crossings in a bit interval T_B = 1 μs is calculated, the difference with the zero-crossing number in the previous T_B is done, and so a bit sequence is estimated; for every signal, two sequences are estimated, one starting with a "0", and the other starting with a "1"
- A comparison of the estimated sequence with the sequences of ID packet and access code is done, and, if they matches, the Bluetooth signal is recognized

First of all, the optimum threshold of the absolute value of the difference of the number of zero crossings was found, in the different conditions of SNR.

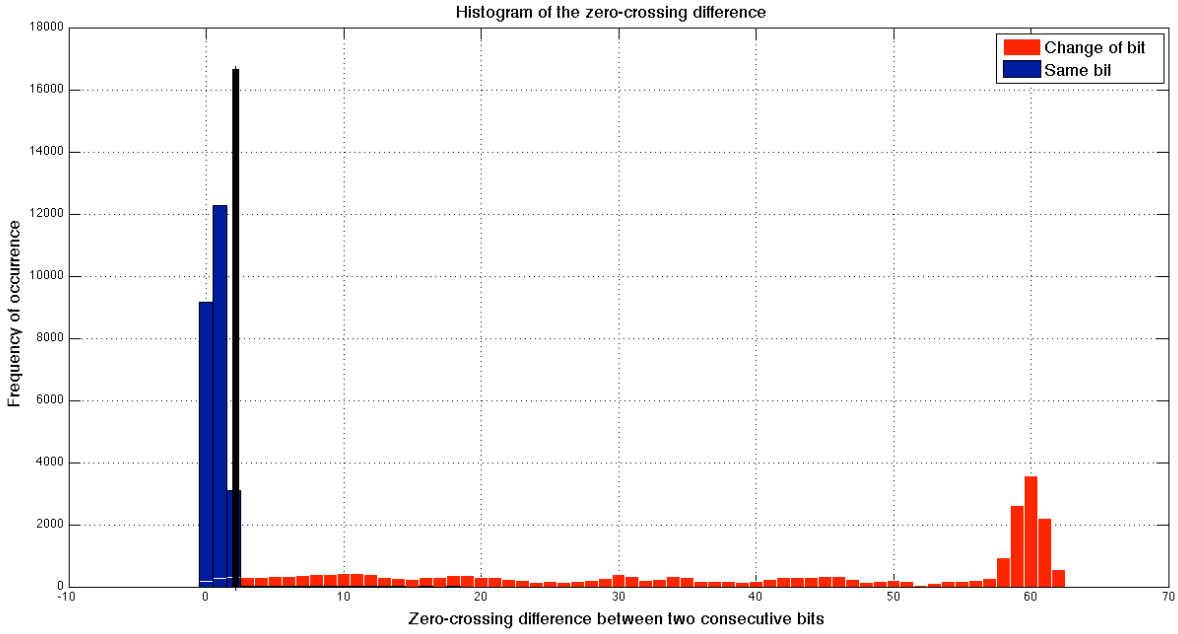

Figure 4.1 – Optimum threshold in zero-crossing, SNR = 40 dB

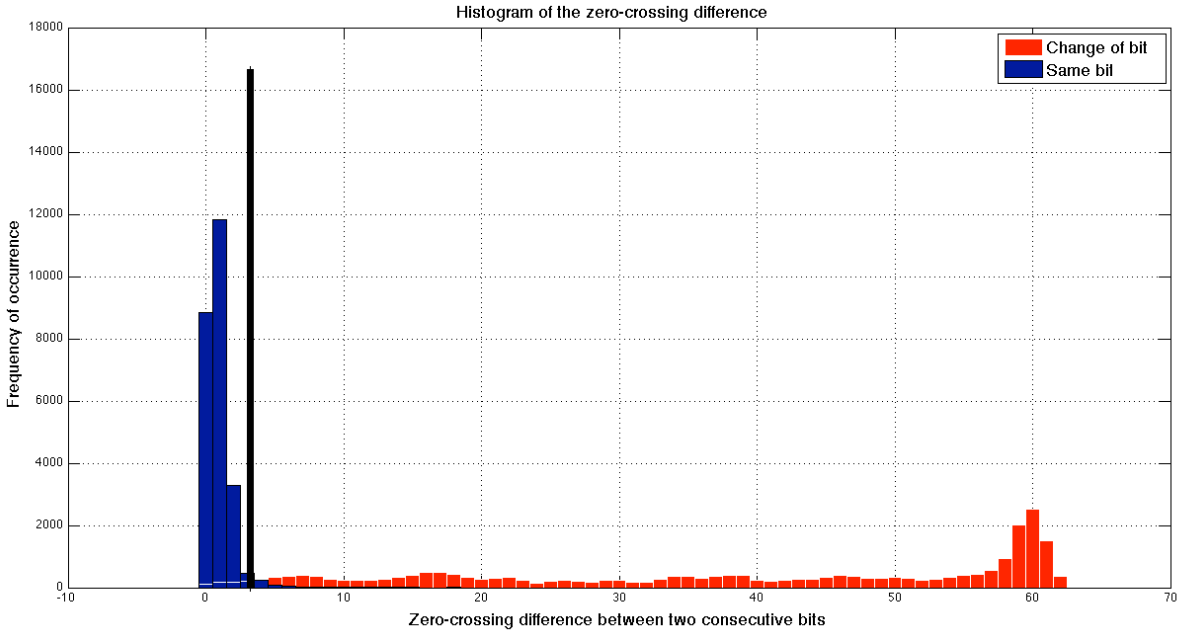

Figure 4.2 – Optimum threshold in zero-crossing, SNR = 20 dB

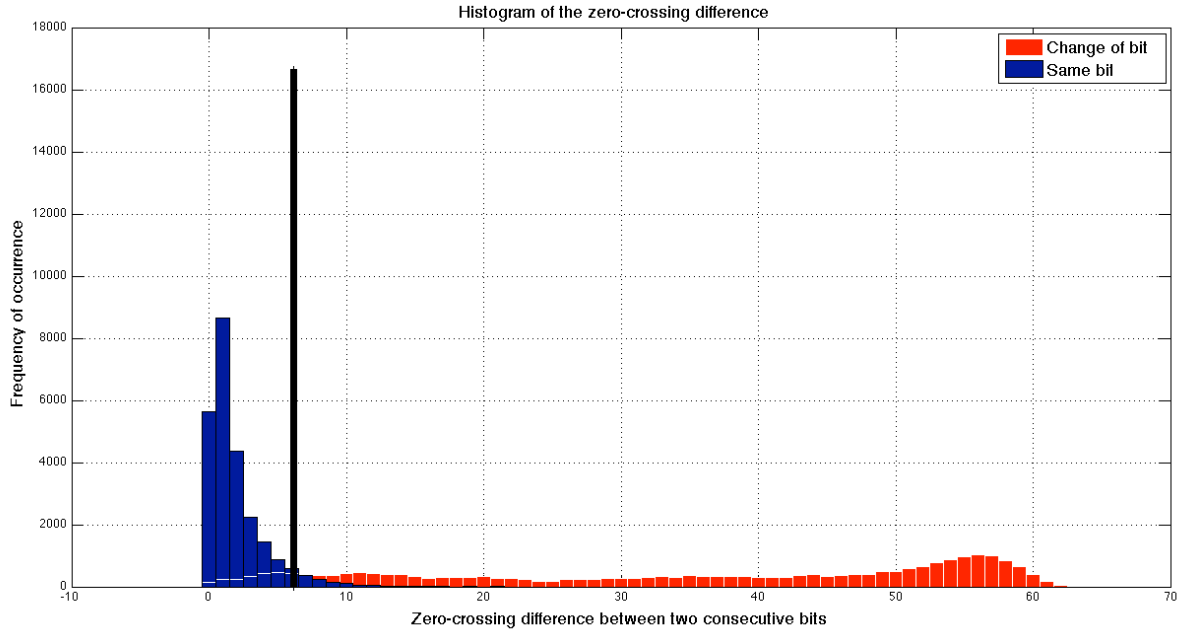

Figure 4.3 – Optimum threshold in zero-crossing, SNR = 10 dB

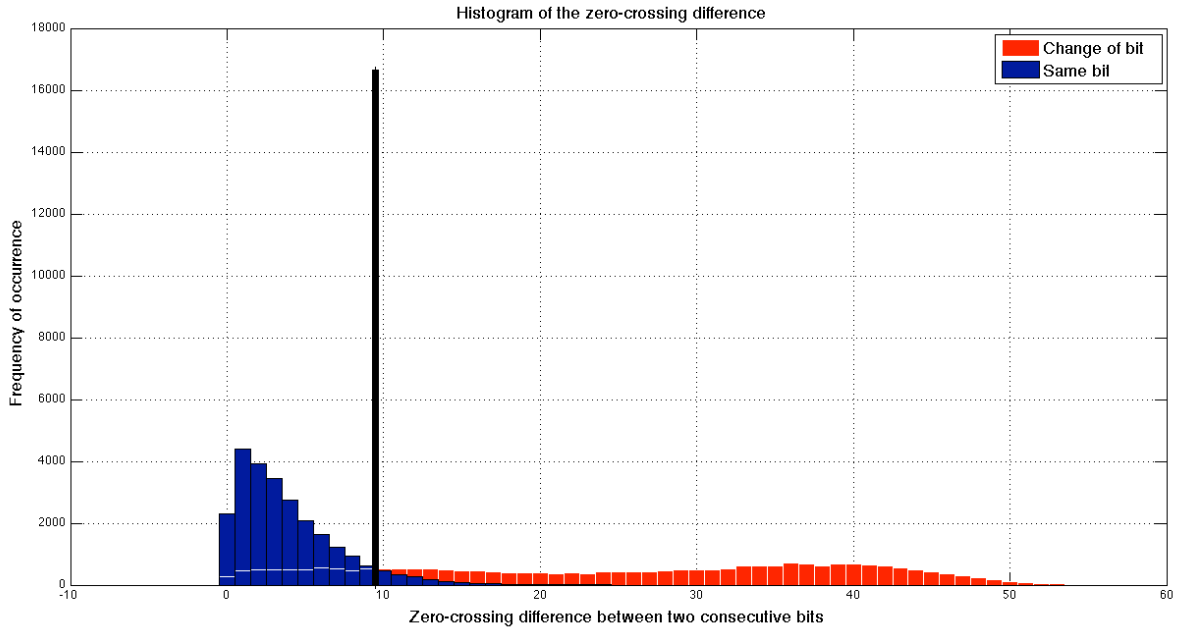

Figure 4.4 – Optimum threshold in zero-crossing, SNR = 5 dB

As expected, when the SNR decreases, the threshold grows, because the noise can increase or decrease the number of zero crossings of the signal, even if it represents the same bit as the previous one.

In fact it can be seen that the histogram of the zero-crossing difference in the "same bit condition" is concentrated near the zero with higher SNR values, while it is more spreaded with lower SNR values.
An analogue situation occurs for the zero-crossing difference in the "change of bit condition", but the concentration is this time near higher values, and it presents a more distributed histogram. The reason of this more distributed histogram, even with an SNR value of 40 dB, is the fact that different carrier frequencies are taken into account. With higher frequencies, the deviation from the centre frequency has less influence, and the "change of bit condition" records a lower absolute value of the difference of the number of zero crossings.

After founding the zero-crossing threshold, the rate of recognized Bluetooth packets (and of the recognized variable-envelope signals) is considered, as the SNR value changes.

Three different scenarios are taken into account:

1. There is no active Bluetooth device in the environment
2. There is (at least) an active Bluetooth device in the environment, but no active Bluetooth connections
3. There is (at least) an active Bluetooth connection in the environment

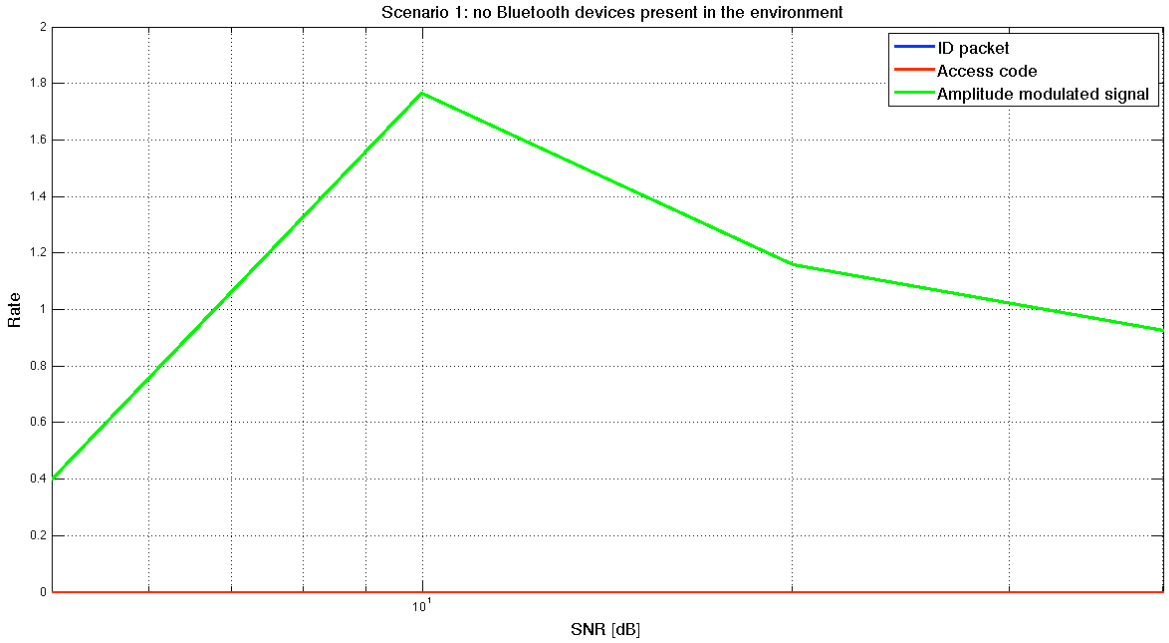

Figure 4.5 – Rates in scenario 1

In the first scenario the rate of recognized ID packets and access codes is, of course, null.

The rate of recognized variable-envelope signals first increases and then decreases, tending at a value near 1 as the SNR grows.

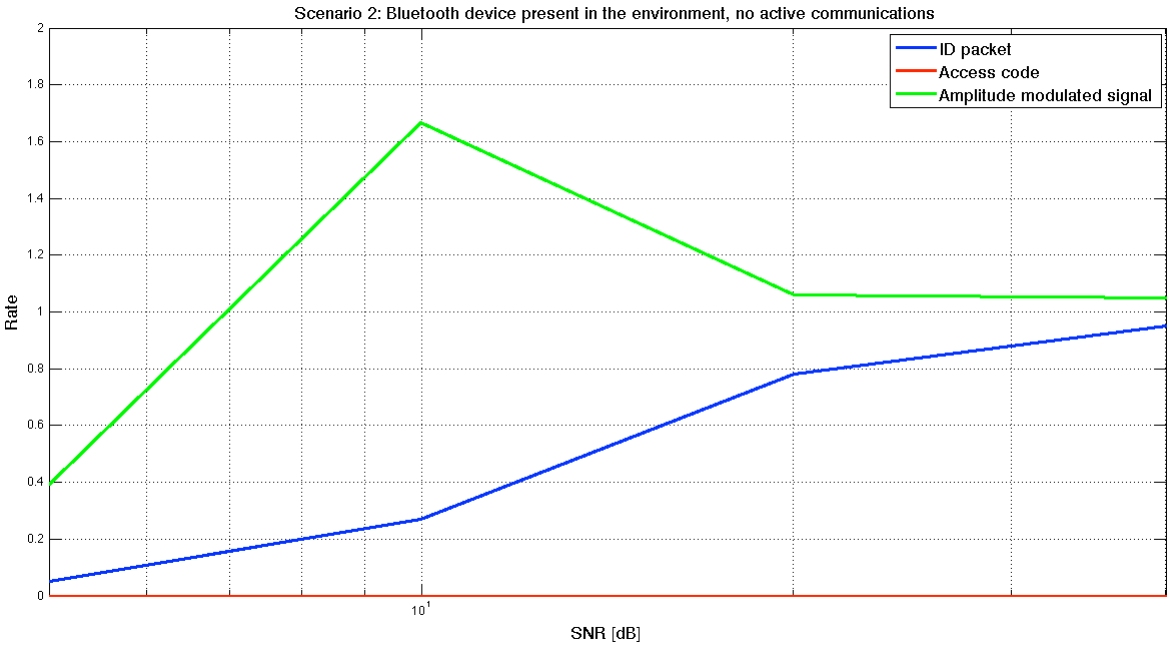

Figure 4.6 – Rates in scenario 2

In the second scenario the rate of recognized access codes remains null (there are no access codes) with every SNR value.

The rate of recognized ID packets increases as the SNR value increases. With SNR = 5 dB its value is near to 0, making it difficult a recognition of the presence of a Bluetooth device (if no bit errors in the bit sequence are allowed). With SNR = 40 dB the rate tends to 1.

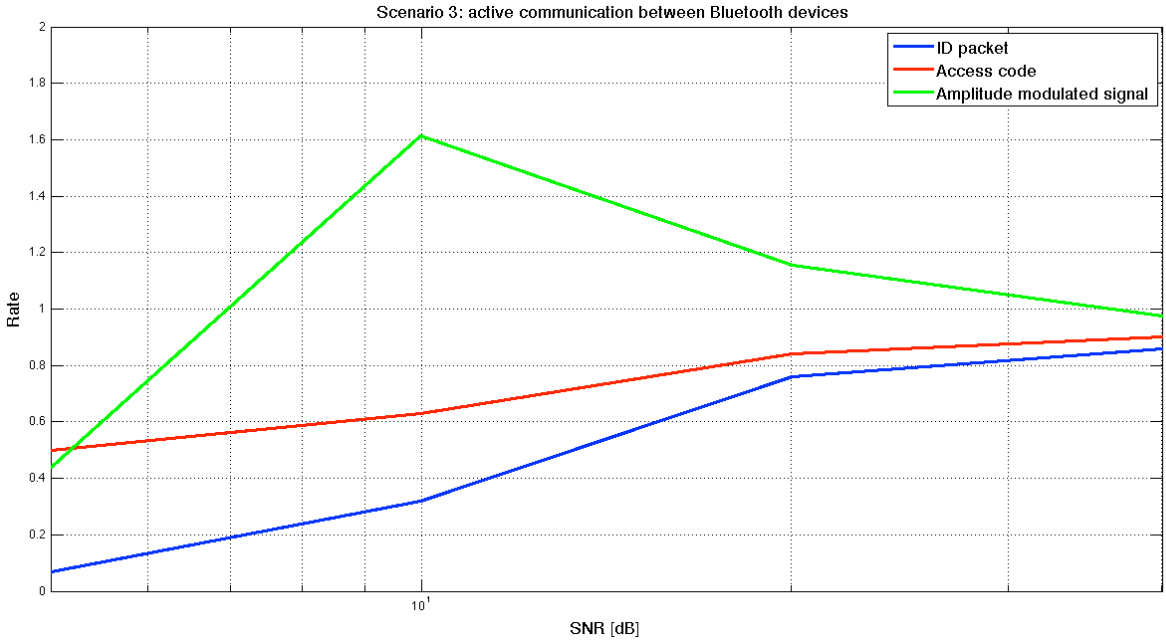

Figure 4.7 – Rates in scenario 3

In the third scenario the rate of recognized access codes can be considered. Like the rate of ID packets, it increases as the SNR value increases, tending to 1 with SNR = 40 dB. But even with SNR = 5 dB, its value does not go under 0.5, making it possible to recognize a Bluetooth active connection. This is due to the lower number of bits that can be compared.

They are in fact only 16 (5 + 11), but in fixed time positions, and they must occur every 625 μs (with 1-time slot packets), so they can be verified frequently. For this reason they can lead to a quite sure answer on the presence of a Bluetooth connection in the environment.

After considering the rates as the SNR value changes, a temporal transition among the threee different scenarios has been taken into account.

```
Number of ID packets present in the environment: 0
Number of access codes present in the environment: 0
Number of sequences modulated ASK present in the environment: 20

No ID packet found

No access code found

Number of amplitude modulated signals found: 21 (rate: 1.05)


Number of ID packets present in the environment: 10
Number of access codes present in the environment: 0
Number of sequences modulated ASK present in the environment: 20

Number of ID packets found: 10 (rate: 1)
There is a Bluetooth device in the environment

No access code found

Number of amplitude modulated signals found: 22 (rate: 1.1)


Number of ID packets present in the environment: 10
Number of access codes present in the environment: 0
Number of sequences modulated ASK present in the environment: 20

Number of ID packets found: 10 (rate: 1)
There is a Bluetooth device in the environment

No access code found

Number of amplitude modulated signals found: 18 (rate: 0.9)


Number of ID packets present in the environment: 10
Number of access codes present in the environment: 10
Number of sequences modulated ASK present in the environment: 20

Number of ID packets found: 8 (rate: 0.8)
There is a Bluetooth device in the environment

Number of access codes found: 10 (rate: 1)
There is an active Bluetooth communication in the environment

Number of amplitude modulated signals found: 24 (rate: 1.2)
```

Figure 4.8 – Change of scenario and the recognized packets with SNR = 40 dB

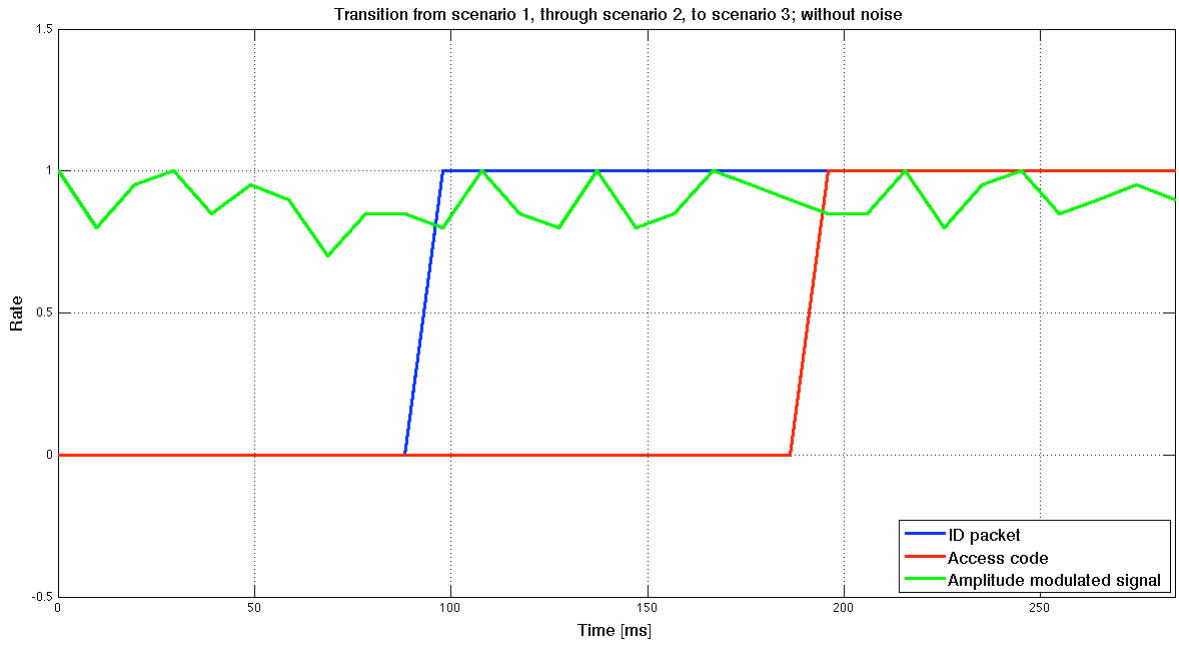

Figure 4.9 – Transition through scenarios, without noise

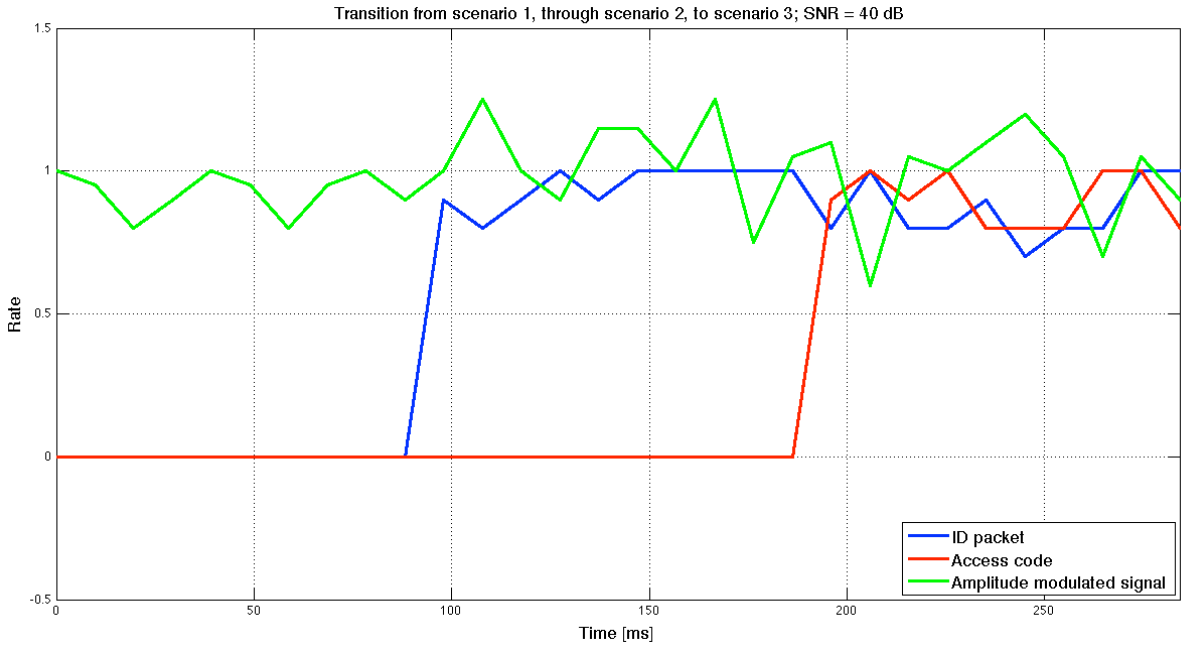

Figure 4.10 – Transition through scenarios, SNR = 40 dB

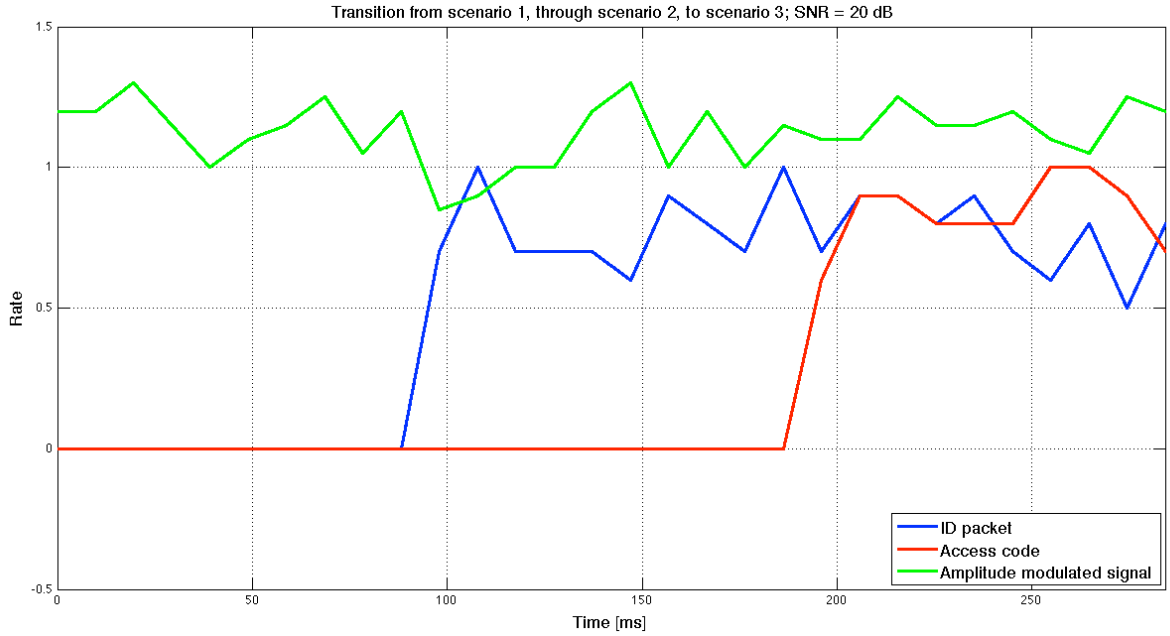

Figure 4.11 – Transition through scenarios, SNR = 20 dB

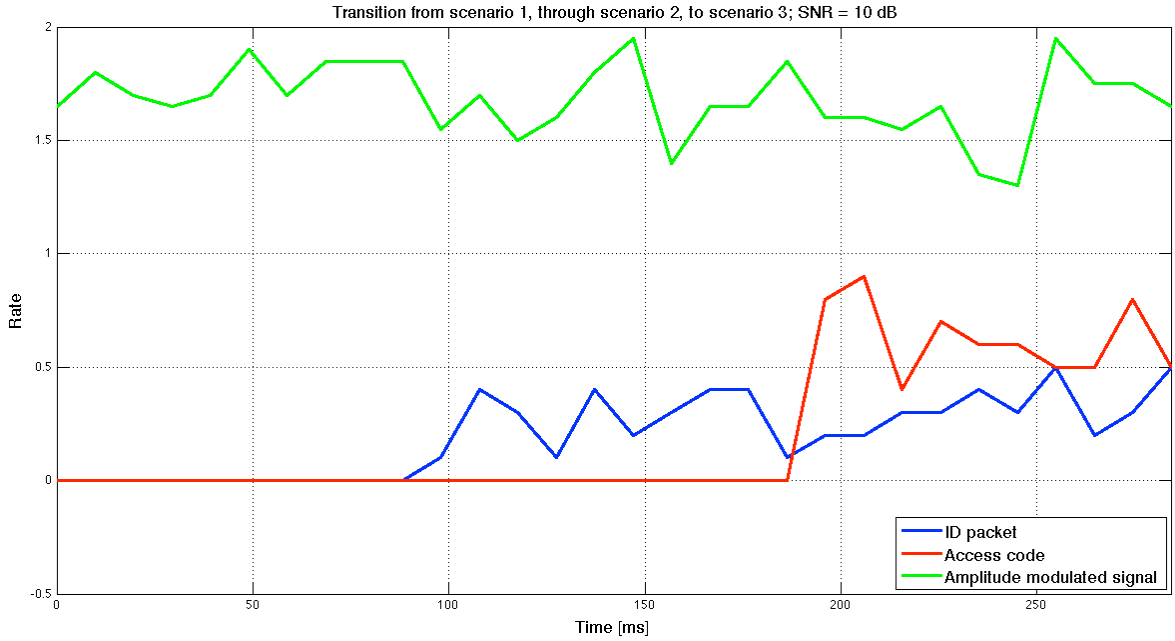

Figure 4.12 – Transition through scenarios, SNR = 10 dB

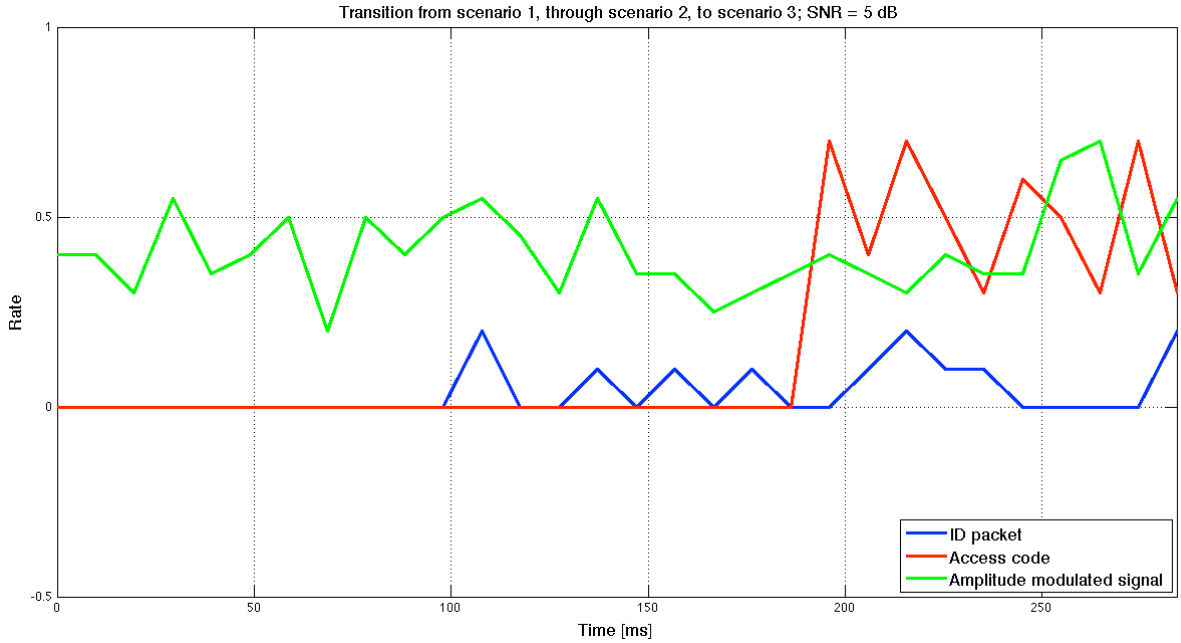

Figure 4.13 – Transition through scenarios, SNR = 5 dB

As it can be seen, the ID packet and access code rates mantain quite high values until the SNR is above or equal the value of 20 dB.

With SNR = 10 dB both rates present lower values.

With SNR = 5 dB the ID packet rate is too low to be recognized. Maybe allowing some bit errors in the 68-bit sequence, it can grow to higher values. The access code rate value remains a little higher, due to its shorter known bit sequence.

CHAPTER 5
CAPTURING REAL TRAFFIC

5.1: RAW mode and Bluetooth sniffer

So far it has been considered simulated traffic: Bluetooth packets and signals have been created, using MATLAB, and they have been analyzed.

It would be interesting to analyze real Bluetooth traffic, as it can be done with the Wi-Fi technology [17] [18].

5.1.1: Problems in Bluetooth sniffing

In order to capture real Bluetooth packets, some difficulties come out.
We can understand them better if we have clear the operating scenario. Our ideal scenario should be this one: two or more Bluetooth devices (one master and one or more slaves, or even more masters if there is a scatternet or just more piconets) which are sending and receiving packets; our Bluetooth device, that can collect all these packets, without being part of the piconet (or the piconets). Our device should be, in other words, a "sniffer".

Here the first problem comes out: capture all the packets.
As we have seen in the Bluetooth technology description, a device accepts a packet only if it matches with its address, or if it is a broadcast packet. If it recognizes the packet is not directed to itself, it automatically discards it. Bluetooth devices are not designed to accept all the packets, this "sniffer function" is not expected.
For this reason, if that function should be obtained, a specific device, designed for this scope, should be used.

Another possible way is to use a normal, everyday Bluetooth device, but with a modified firmware, that allows to use it as a sniffer.

Another difficulty in creating a Bluetooth sniffer resides in the Bluetooth technology itself, as it was thought and designed.
It has been seen in chapter 2 that it uses the Frequency Hopping Spread Spectrum to avoid interference and fading. Packets are spread over 79 frequencies, and the link hops between these, changing frequency for every packet. Naturally a device can be tuned to only one frequency at a time.
So, if Bluetooth packets shall be captured, the device has to know to which channel it has to listen to in every instant. In other words it has to know the hopping sequence, that is not a trivial thing.
Then, if we manage to solve this problem, we have to consider that in this way only the traffic of the piconet related to the hopping sequence can be captured, and not the traffic of the other piconets eventually present in the area.

The realistic scenario, in which real Bluetooth traffic capture can be done, will be, therefore, different from the ideal one.
With a Bluetooth sniffer only the packets of one piconet can be obtained, and only if the device is able to know, somehow, the hopping sequence.

5.1.2: RAW mode

The first thing is, as we have seen, to obtain a Bluetooth sniffer.

The first hypothesis is to use a specific device. An example of this is the FrontLine FTS4BT, that is a protocol analyzer and packet sniffer. It is designed to capture packets, so it can solve our problem.
But using a specific device turns us away from our primary target, that is the cognitive radio. If a generic cognitive radio device, that can recognize wireless networks eventually present, will later be used, a specific device can not be used to

capture real traffic, because this device will not be available in the final implementation.

For this reason the second hypothesis must be chosen: a normal Bluetooth device, but modified for our scope.
To obtain a sniffer, our device should not discard packets not directed to itself, but it should collect all the packets.
In the Wi-Fi technology, a similar behaviour is obtained putting the Network Wireless Adapter in the so called “monitor mode”. After doing that, it collects all the Wi-Fi packets in a specific Wi-Fi frequency channel [17] [18].

For a Bluetooth device there is the “RAW mode”, that allows it to capture all the packets in the frequency channel to which is tuned to.
Therefore our device should be put in RAW mode. This is not immediate, because this is not an expected mode for a device. The device's firmware must be changed with another firmware that allows the RAW mode.

First of all some hardware limitations come out: it is not possible to do this change in all the chipsets of the devices. A suitable chipset shall be found (in this work a Cambridge Silicon Radio chipset has been used), and then a firmware that puts the device in RAW mode. The device's firmware must be changed with this new one, and we finally get a Bluetooth device in RAW mode: our sniffer.

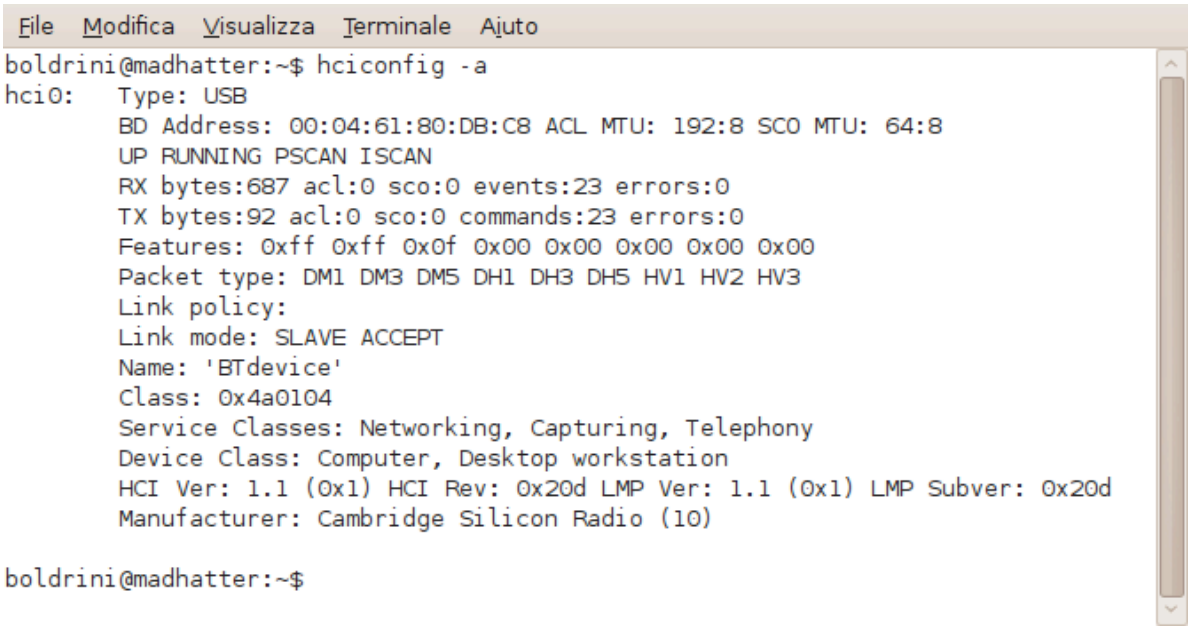

Figure 5.1 – Bluetooth device before turning it in RAW mode

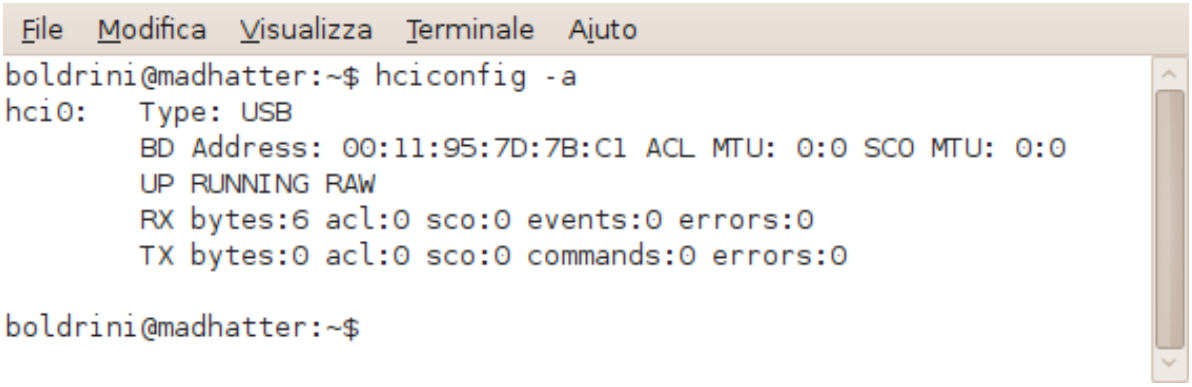

Figure 5.2 – Bluetooth device after turning it in RAW mode

5.1.3: Association to a piconet and data acquisition

Now the device can be controlled with some special commands, that depend on the firmware that has been used.

To use our sniffer to capture the Bluetooth packets, it has to know the hopping sequence. To do that, it has to be specified, through the opportune command, the BD_ADDR of the piconet's master. With this address, our device extracts its clock and calculates the corret hopping sequence.

Now our sniffer is able to capture all the packets sent in the piconet. Thanks to this special firmware it does not discard packets not directed to itself, and thanks to the hopping sequence it knows in which channel they are transmitted in every instant.

It should be noted that in this way the sniffer knows the hopping sequence, but it is not associated to the piconet. That's corret for a sniffer.

At this point an opportune application shall be used, to obtain the packets, the sniffer has captured, from the sniffer itself.
But before doing this, it must be verified if this is really possible, with our Bluetooth device.

5.2: Host Controller Interface

Before capturing real Bluetooth packets, it must be considered an important part of the Bluetooth technology described in the IEEE Standard 802.15.1 – 2005: the Host Controller Interface (HCI).
As the name says, the HCI is a standardized interface which provides a control system of the Bluetooth device from the higher architectural layers of the host.

The figure below shows the Bluetooth protocol stack, and where the HCI takes place in it:

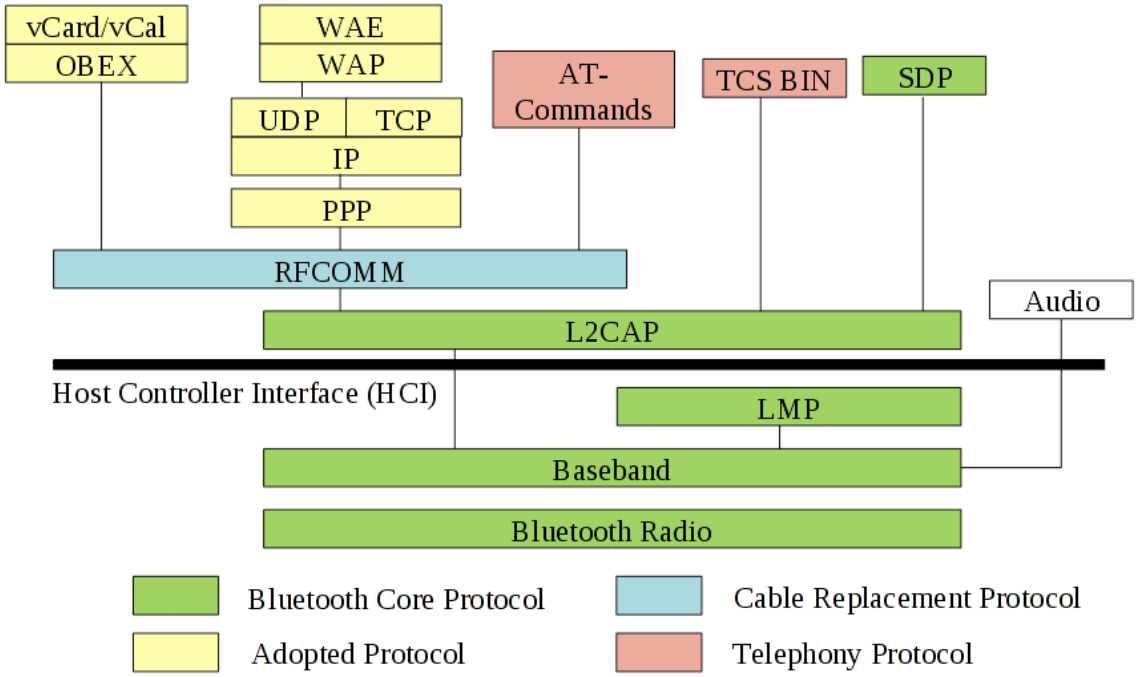

Figure 5.3 – Position of the Host Controller Interface

The HCI is positioned between the BaseBand (BB) layer and the Logical Link Control Adaptation Protocol (L2CAP) layer.

5.2.1: Why HCI

The function of the HCI is to interface the host with the Bluetooth device, to create a link between them, so they can communicate in a correct way and inter-operate with each other.

In fact the host is intended to be generic, that means it can be a cellular phone, a smartphone, a netbook, a computer, or even another device. Naturally every kind of device has its own specific functional rules and protocols, that can be different from the rules and protocols used in other kind of devices.
On the other hand a Bluetooth device can be developped and manufacted by a very large number of manufacturers; and every device has its own specific firmware. To

say more, in general it is not known how the firmware exactly works, how the functions are effectively implemented (except of the manufacturer, of course).

That is why an interface is needed. The HCI fills this gap between this two entities of different type, between the various nature of the host device and the proprietary implementation of the Bluetooth device's firmware.
The HCI specifies how the upper layers can control the device and which functions and parameters they shall use to communicate with it.
On the other side, through HCI, the device knows how to interpret the upper layers commands, and perform the correct operations.

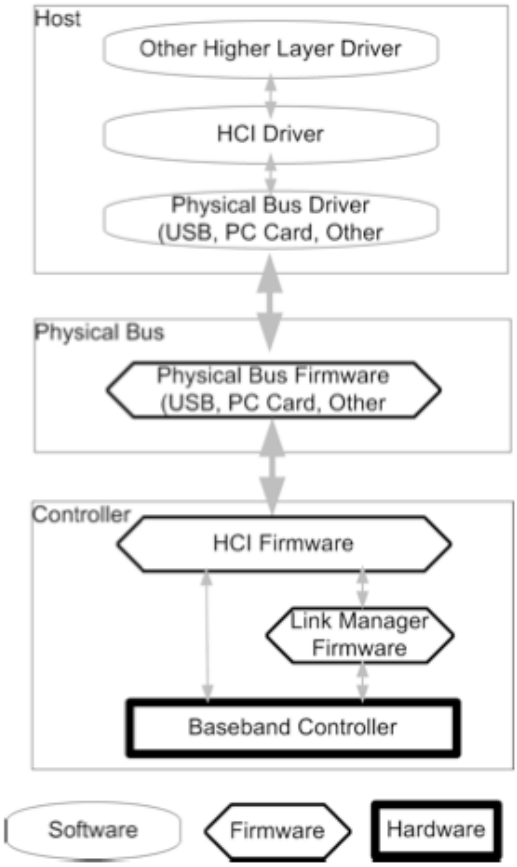

Figure 5.4 – HCI: between host and device

The IEEE Standard specifies all the possible commands supported by the HCI, so that every host device and every Bluetooth device can implement them and communicate with the "other part" through the interface in a correct way.

5.2.2: Limitations imposed by HCI

It has been seen why the Host Controller Interface has been introduced and what are its functions.
For the scope of this work, however, the presence of HCI represents a sort of limitation.

As it can bee seen, the Physical Radio layer and the BaseBand layer remain under the HCI. This allows the manufacturer of the Bluetooth device to implement its own firmware, with the only fixed point to respect the IEEE Standard specifications.
But this HCI position also implies that every communication between lower layers, that is to say not derivated from higher layers commands, remains under the HCI, and any host can not have any kind of control on it.

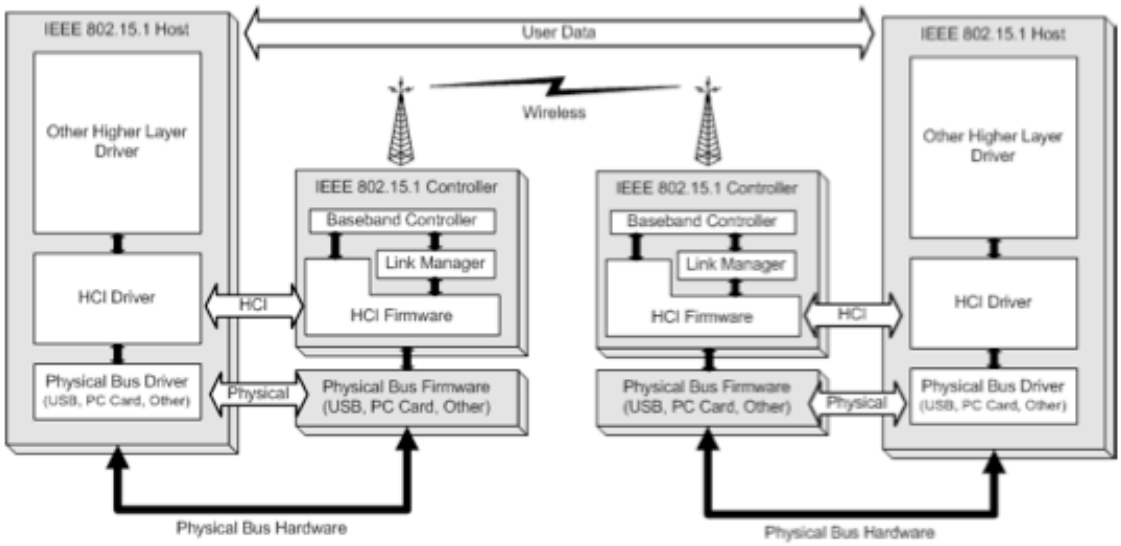

Figure 5.5 – Communication between layers of the same level

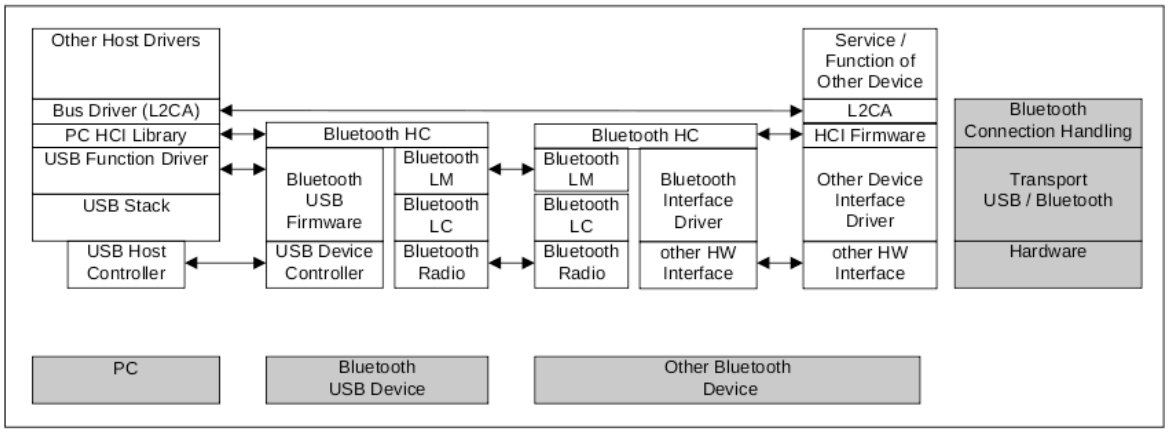

Figure 5.6 – Protocol stack from PC to Bluetooth device

For example, if a host wants to send to another host some "user data", i.e. data generated in higher layers of the stack, these data are encapsulated in lower layers packets, pass through the HCI, are encapsulated in the lowest layers packets and are sent. At the receiver side the inverse operations are done, to receive the user data.
In this communication all the data created in the layers that reside above the HCI are available at the receiver host, both payload and header of the different layers packets.

However a control packet does not cross the interface. This kind of packet, that is generated from the layers below the HCI, follows the normal path of encapsulation in lower layers packets, but at the receiver side it goes up until it reaches the layer from which it was generated, that is obviously even in this stack below the Host Controller Interface, not crossing it.

In other words, all the data present in the host are accessible, but the data present in the host are the ones generated from the architectural layers that reside in the host side, above the HCI, and not the ones generated from the architectural layers in the Bluetooth device side, below the HCI.
The lower layers packets, as the control packets, are therefore unreachable, because they do not cross the HCI and always remain in the device side.

To be more precise there could be a way to reach these packets: changing the firmware of the Bluetooth device with another one specifically designed for this

scope, so that it can include functions to obtain these data. This solution has not been proved in this work.

5.3: Inquiry packets not available

For the reasons explained above, it is not possible to obtain control packets from a normal Bluetooth device. Unless a change of firmware with a specifically modified one, or the use of a dedicated device, these kind of packets are not available, because they do not cross the Host Controller Interface.

This is the case of the inquiry packets. As explained, they could be really interesting thanks to their temporal regularities. But they are control packets, and it has not been possible, in this work, to obtain real Bluetooth inquiry packets with our device.

CHAPTER 6
CONCLUSIONS AND FUTURE WORKS

6.1: Conclusions

In this work the Bluetooth technology has been studied, and some characterizing features have been found.
A Bluetooth transmitter simulator has been created using MATLAB, and Bluetooth packets and signals have been generated. They have been analyzed, and Bluetooth sequences have been distinguished from the the others using the features.
In this way the recognition of Bluetooth signals based on feature detection has been reached.

From the results graphs it can be seen that the Bluetooth recognition is possible using the proposed features, and without knowing the istantaneous carrier frequency f_c.
With high values of SNR the recognition is supported by rate values near to 1, and it can be considered "sure" with a certain reliability.
When the SNR becomes low, the rate of recognized sequences gets lower. In the case of access codes, the rate still remains above the 0.5 value. In the case of ID packet, because of the longer sequence and without permission of bit errors, the rate decreases to lower values.

From these results, the proposed features seem valid for Bluetooth recognition.
With low values of SNR, either a longer acquisition can be provided, or some bit errors should be permitted.
With a longer acquisition, even if the rate remains low, a growing number if sequences is found, and that is very unlikely if no Bluetooth device is present in the environment.
Permitting some errors on bit sequences, especially on the ID packet, that is 68 bits long, can make the rate higher, even if some errors could occur.

A brief analysis on higher layers features shows that they can distinguish Bluetooth from Wi-Fi, even if no strong regularities permit Bluetooth identification itself.

6.2: Future works

It could be interesting to prove the obtained results implementing this simulation in the hypothesized generic device; this could be an important element in the cognitive radio device.

Overpassing the Host Controller Interface, a real data analysis can be done, and some other higher layers features can be found, that could permit Bluetooth recognition without the need of using the physical layer.

APPENDIX

AUTOMATIC NETWORK RECOGNITION BY FEATURE EXTRACTION: A CASE STUDY IN THE ISM BAND

Accepted for publication in *Proceedings of the 5th International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, Special Session on Cognitive Radio and Networking for Cooperative Coexistence of Heterogeneous Wireless Networks, June 9-11 2010, Cannes, France.

Automatic network recognition by feature extraction: a case study in the ISM band

Maria-Gabriella Di Benedetto, *Senior Member, IEEE*, Stefano Boldrini, Carmen Juana Martin Martin, and Jesus Roldan Diaz

***Abstract*— Automatic network recognition offers a promising framework for the integration of the cognitive concept at the network layer. This work addresses the problem of automatic classification of technologies operating in the ISM band, with particular focus on Wi-Fi vs. Bluetooth recognition. The proposed classifier is based on feature extraction related to time-varying patterns of packet sequences, i.e. MAC layer procedures, and adopts different linear classification algorithms. Results of classification confirmed the ability to reveal both technologies based on Mac layer feature identification.**

***Index Terms*—Cognitive networking, network discovery, automatic network classification**

I. Introduction

This work is framed under the umbrella of the AIR-AWARE Project, developed to achieve classification amongst technologies and interference entities operating over the ISM band. This project aims at creating a black box — the AIR-AWARE module — capable of classifying technologies, as well as different types of interference in play.

Such classification is important for cognitive mechanisms to be implemented in the network, given the numerous commercial technologies operate in this range of the spectrum, such as:

- IEEE 802.11 networks: 2.4 *GHz* and 5.8 *GHz* bands;
- Bluetooth: 2.4 *GHz* band, using Frequency Hopping and any of 79 available channels;
- HIPERLAN [High Performance Radio LAN]: European alternative to IEEE 802.11, operating in the 5.8 *GHz* band to avoid interference entities at 2.4 *GHz;*
- Closed-Circuit TV: Security cameras at 2.4 *GHz*;
- ZigBee IEEE 802.15.4: 2.4 *GHz* range band;
- Wireless Mouse and Keyboard: 2.4 *GHz* band.

Probably, the most common interference at 2.4 *GHz* comes from the microwave oven, followed by baby monitors and cordless Wi-Fi phones, and the interference produced by DECT standard cordless phones, operating in the 1.9 *GHz* band. When in use, these devices can compromise the quality of an IEEE 802.11 network.

The final goal is to propose a classification strategy based on information regarding protocol layers above the physical one (PHY). In particular, the objective is to identify MAC sublayer [1,2] specific features for each of the above technologies. Previous work, as for example [11], has addressed a similar problem, by classifying Wi-Fi vs. Bluetooth, using a spectrum sensing procedure based on distributed detection theory. The present work extends beyond previous investigations by considering Wi-Fi real traffic captures, and by focusing feature extraction and classification on MAC sublayer characteristics, leading to simplicity and computational efficiency.

In this work, feature identification lays its foundation on the observation that packet interchange patterns are technology-specific. As such, by identifying patterns clues, network recognition can be achieved. In particular, the study focuses on the ISM 2.4 *GHz* band. A first step consists in providing the AIR-AWARE module with a device capable of sensing the spectrum with a good time resolution. Albeit this piece of hardware will not be in a position to demodulate and decode the distinct signals in the air ; it will enable AIR-AWARE to statistically study temporization of presence or absence of energy - against pre-defined thresholds – and therefore decode the packet sequence structure, in real time. Figure 1 illustrates the schematic of the cognitive energy detector, as well as software modules for the recognition each technology embedded onto the device.

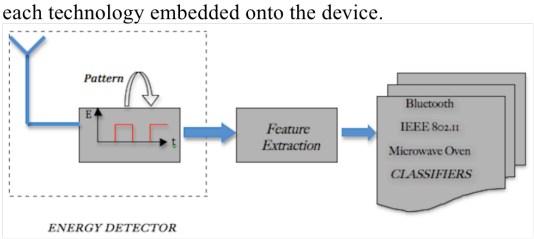

Figure 1 - The AIR-AWARE module

The study presented in this paper focuses on 802.11 (Wi-Fi) and 802.15.1 (Bluetooth) network recognition. To achieve recognition of both technologies, a set of features will be proposed. These features will serve as input for a linear classifier, that automatically performs classification among these two technologies.

The paper is organized as follows. Section II contains the description of the experimental set-up, that is of how the environment and tools by which data were collected and generated. The packet data-base is described in Section III, and particular focus on the Wi-Fi data-base (Section III.A) vs, the Bluetooth data-base (Section III.B). The classification algorithms are all linear classifiers as analyzed in Section IV; four different approaches are taken into consideration, i.e. Perceptron, Pocket, Least Mean Squares (LMS), and Sum of Error Squares Estimation (SOE). Experimental results on automatic classification of Wi-Fi vs. Bluetooth are reported in Section V. These results are discussed in Section VI, which also contains the conclusions of this study.

Manuscript submitted March 21, 2010. This work was supported in part by the European Commission in the framework of COST Action IC0902: Cognitive Radio and Networking for Cooperative Coexistence of Heterogeneous Wireless Networks.

The authors are with the Info-Com Department, School of Engineering, University of Rome La Sapienza, Via Eudossiana 18, 00184, Rome, Italy.

E-mail address: dibenedetto@newyork.ing.uniroma1.it.

II. Experimental set-up

This Section describes the environment and tools used for collecting the reference data and build-up the data-base.

As for Wi-Fi, real data packets were detected within experimental measurements, using a packet capturing device ("Sniffer Station"). These measurements were carried out in the ACTS laboratory, located on the 2nd floor of the Information and Communication Department (InfoCom Dept.), in the Faculty of Engineering of the University of Rome "La Sapienza", Rome, Italy.

A long corridor, with offices and laboratories on both sides, composes the 2nd floor of the building, the ACTS lab being one of these laboratories. The (main) Access Point, to whose frequency channel the sniffer was tuned to, is located in this corridor, near the ceiling, at a height of about 2.5 - 3 *meters* from the ground, and at a distance of about 4 - 5 *meters* from the ACTS laboratory door.

The computers were placed on a fixed location on a table, at a height of about 1 - 1.5 *meters* from the ground, at a distance of about 1.5 *meters* from the nearest wall and of 2 *meters* from a door, that leads to the corridor mentioned above, i.e. the distance between the computers and the Access Point was about 5 - 6 *meters*. The distance between each computer was about 0.5 *meters*. On this same floor there are about 20 additional offices, containing about 2 - 3 computers each. Since we expected that any of these may be connected to the main Access Point during the measurements, and in the aim of capturing "clean data" for the training set, we performed measurements at night after verification that no other computer was active during captures.

There are also other Access Points in the same building, but tuned to other frequency channels. However, some packets of these other Access Points may have been captured. That occurs because of the frequency channels partial overlapping, that is expected in the IEEE 802.11 Standard. To this respect, collected data were carefully examined, in order to ensure again that only packets of the relevant network were captured.

Four computers were used to perform measurements. Three computers were used to generate traffic in heterogeneous traffic conditions. The fourth, used as the Sniffer Station, had the following technical specifications:

- Model: *HP Pavilion DV2320US*
- Processor: *AMD Turion 64 X2 TL 56*
- RAM Memory: 2*GB*
- Network Wireless Adapter: *AirForce 54g 802.11 a/b/g PCI Express*, with a *BCM4311* chipset

Operating System was *Linux Ubuntu 9.10*, with the real-time kernel *2.6.31-9-rt*. The driver used for the Broadcom 4311 chipset was the b43. The main Access Point was a *Cisco Aironet 12xx 802.11 b*.

As for Bluetooth packets, in this first phase we decided to design a complete simulator by which Bluetooth packets were obtained. By doing so, one of the two data packet stream was fully controllable by software.

III. Packet data-base

A. Capturing Wi-Fi packets

In order to capture Wi-Fi Packets, a packet capturing application was developed using the Java library *jpcap* [7]. The Wi-Fi standard foresees a logic unit, the MAC PDU, while the unit sent over the air interface, called PPDU, includes beyond the MAC PDU, two additional fields (preamble and header). The driver used for the 802.11 network adapter enabled to intercept data of every PPDU [1] within the Sniffer range by means of its monitor mode [8]. This driver was also compatible with the radiotap header [9], which provides information such as preamble type, or time of arrival of first bit, for the captured MAC PDU.

Experiments were made in three different conditions: one, two, and three computers (nodes) associated to the Access Point. In all conditions, each computer was downloading at least two files or processing a video call; this way, we could secure a high traffic scenario.

Two 1000-packet captures were run for each condition.

B. Generation of Bluetooth packets

Bluetooth simulated packets were generated using MATLAB. The reference standard for this simulation is the IEEE Standard 802.15.1 – 2005 [2], i.e. bitrate of 1 *Mbit/s*. Piconets of two devices in connection state (one master and one slave) were considered. The two devices send their packets alternately: one device (the master, for example) sends its data packets, and for every received packet, the other device (the slave) sends back an acknowledgement. Data packets sent by the master can occupy 1, 3 or 5 time slots (where the time slot is 625μs), according to their length, whereas acknowledgement packets (NULL packets, with a fixed length of 126 *bits*) occupy 1 time slot.

Two different scenarios were considered:

- Scenario 1: data packets occupy only 1 Time Slot
- Scenario 2:
  - 80% Data Packets occupy 1 Time Slot
  - 15% Data Packets occupy 3 Time Slots
  - 5% Data Packets occupy 5 Time Slots

In every scenario, 70 % of the data packets have a duration that is fixed by the protocol to the values shown in Table I. The duration of the remaining 30% is uniformly distributed between minimum and maximum values (see Table I).

According to the standard, for every packet arrival time a jitter of ± 10μs has been set, to consider imperfect synchronization between the two devices. The jitter was modeled by a Gaussian distribution with zero mean and standard deviation σ=10/3μs; given the model, 99% of jitter values fell within a ± 10μs interval, while the remaining 1% exceeded this interval and were readjusted in order to meet the standard specifications.

TABLE I
BLUETOOTH STANDARD SPECIFICATION

	Fixed duration	Min. Duration	Max. Duration
Time slot	625μs		
1-time-slot-packet		126μs	366μs
3-time-slot-packet		1250μs	1622μs
5-time-slot-packet		2500μs	2870μs
NULL packet	126μs		

IV. AUTOMATIC CLASSIFICATION

We started by designing a generic linear classifier that was able to distinguish amongst *C* classes, each class being characterized by *M* features. In general, a linear classifier divides the feature space into *C* regions; this division comes about through calculation of discrimination functions that characterize the region of the space where each class is located:

$$g_j(x) = w_{0,j} + \sum_{i=1}^{M} w_{i,j} x_i, \qquad j = 1,2,..,C \quad , \tag{1}$$

where $w = [w_0, w_1, ..., w_M]$ is known as the weight vector, and $X = [x_1, x_2, ..., x_M]$ is a point on the decision hyperplane.

The classifier objective is to find each discrimination function, and generate a decision using the major score criterion. This score represents a measure of similarity between the object and each class. The classification module, was implemented on MATLAB, based on four classification methods selected because of their simplicity and computational appeal:

- Perceptron. One of the oldest methods, its convergence depends on the separability of the classes. The idea is to calculate the weight vector through an iterative method, formulated in this way [3]:

$$w(t+1) = w(t) - \rho_t \frac{\partial J(w)}{\partial w}\Big|_{w=w(t)} \quad , \tag{2}$$

where, as displayed in the equation, there is a learning coefficient ρ_t and the minimization of a cost function is required, defined in this case as :

$$J(w) = \sum_{X \in Y} \delta_X w^T X \quad , \tag{3}$$

where Y represents the training set, and δX is a coefficient that takes value equal to -1 if X Class1 and equal to 1 if X Class2 or vice-versa. In the multiclass case, δX equal to -1 for all classes.

- Pocket. A version of Perceptron, that provides a better behavior when separability of the classes is not totally guaranteed. The key [6] is to run perceptron learning, while keeping an extra set of weights in the pocket. By this way, if a new iteration provides a weight vector that classifies a greater number of training vectors than its predecessor, then that last is chosen and used in the next step; otherwise, the vector obtained in the preceding iteration is maintained as the weight vector.
- Least Mean Squares Method (LMS). Similar to Perceptron, but the cost function, that is minimized corresponds to the error. Here, the weight vector is computed so as to minimize the Mean Square Error (MSE) between true and desired output (y) [3]:

$$J(w) = E\left[\left|y - X^T w\right|^2\right] \quad , \tag{4}$$

where the mean value (that cannot be generated due to the lack of statistical data), is replaced by samples obtained during experimentation. The weight vector is therefore obtained according to the following rule:

$$w(k) = w(k-1) + \rho_k X_k (y_k - X_k^T w(k-1)) \quad , \tag{5}$$

where ρ_k is a learning coefficient.

- Sum of Errors Squares Estimation (SOE). Similar to LMS, here the cost function takes the form of the sum of quadratic error for each of the N training vectors X:

$$J(w) = \sum_{k=1}^{N} (y_k - X_k^T w)^2 = \sum_{k=1}^{N} e_k^2 \quad , \tag{6}$$

where the calculation of weight vectors is the simple matrix operation [3] shown below:

$$\sum_{k=1}^{N} X_k (y_k - X_k^T w) = 0 \Rightarrow (\sum_{k=1}^{N} X_k X_k^T) w = \sum_{k=1}^{N} (X_k y_k) \tag{7}$$

In order to compute both optimal w and desired values *y*, the Ho-Kashyap algorithm was used [10].

V. EXPERIMENTATION

A. Training set and feature extraction

As described in Section III, in the Wi-Fi case, six (6) 1000-packet captures formed the training set. In the Bluetooth case, simulated MATLAB captures consisted in two 6000-packet sequences corresponding to Scenarios 1 and 2.

The first proposed feature is the time interval between PPDUs, defined in [1] as Short Inter Frame Space (SIFS) corresponding to silence gaps on the medium when DATA-ACK procedures are in play. Most IFS timings are fixed and independent of the bitrate at the PHY [1]. Of all existing IFS types, SIFS has a nominal value of 10μs for the ISM 2.4GHz band, and is the likely to occur in a scenario with medium to high traffic; it is usually used by a node responding to any polling, and always prior to: a) transmission of an ACK frame; b) a CTS frame; c) a second or subsequent PPDU of a fragment burst. For automatically extracting the SIFS and estimating its statistical behavior, SIFS was differentiated from a non-SIFS when two consecutive PPDUs, durations were such that: $0.6*PPDU_{ith} > PPDU_{ith+1}$.

The second proposed feature is the duration of the longest packet considering all the packets between two consecutive silence gaps, previously considered as SIFS.

Note that both proposed features are extremely simple and easy to extract thanks to simplest hardware such as an energy detector.

Figure 2 illustrates the feature plane for both Wi-Fi (real traffic and Bluetooth (simulated traffic) training set. The Bluetooth data correspond to Scenario 1 (single-slot case).

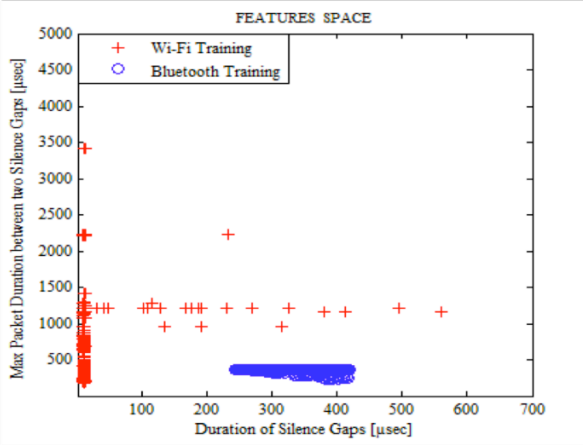

Figure 2 - Feature Plane for Wi-Fi and Bluetooth single-slot

Figure 3 shows the feature plane in the multi-slot Bluetooth Communication scenario (Scenario 2).

Note the presence of a few Wi-Fi points invading the Bluetooth "zone". Capture file revision indicated that these corresponded to non-SIFS, i.e. erroneously estimated SIFS. These points were however less than 1% of total.

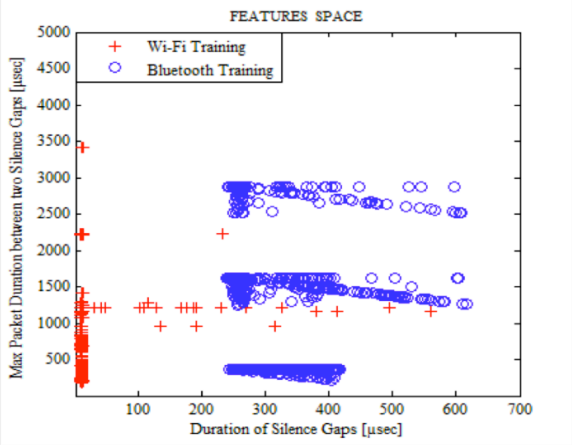

Figure 3 - Feature Plane for Wi-Fi and Bluetooth multi-slot

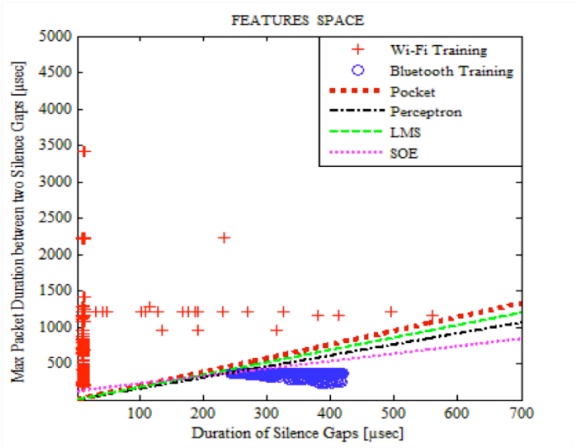

Figure 4 - Automatic classification of Wi-Fi vs. Bluetooth single-slot

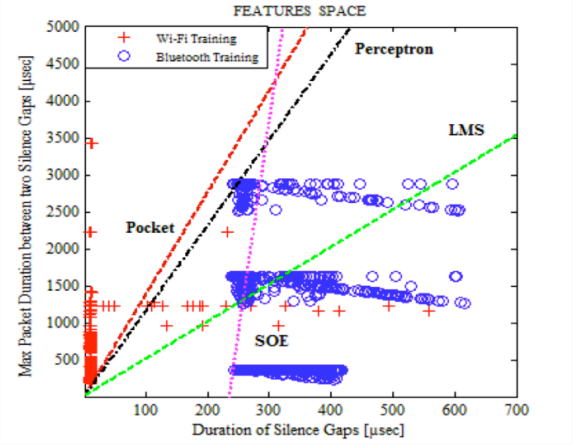

Figure 5 - Automatic classification of Wi-Fi vs. Bluetooth multi-slot

B. Results of automatic classification

The four classification algorithms were run over the training sets of Figs. 2 and 3. Results are reported on Figs. 4 and 5, for the single-slot vs. multi-slot Bluetooth cases. The classifiers were then applied to data not belonging to the training sets, i.e. a new 1000-packet Wi-Fi capture (1.4 seconds capture duration), and two new 1000-packets Bluetooth simulations (Scenarios 1 and 2) were generated (each around 0.7 seconds long). Results of classification percentage of Wi-Fi vs. Bluetooth (single-slot case), when the input to the classifier is formed by either Wi-Fi captures or Bluetooth sequences of packets are reported in Tables II and III, for the single vs. multi-slot Bluetooth, respectively.

TABLE II
CLASSIFICATION RESULTS

	Classifier Input Network	Classification into Wi-Fi	Classification into single-slot Bluetooth
Pocket	Bluetooth	0% [0/456]	100% [456/456]
Pocket	Wi-Fi	100% [352/352]	0% [0/352]
Perceptron	Bluetooth	0% [0/456]	100% [456/456]
Perceptron	Wi-Fi	100% [352/352]	0% [0/352]
LMS	Bluetooth	0% [0/456]	100% [456/456]
LMS	Wi-Fi	100% [352/352]	0% [0/352]
SOE	Bluetooth	0% [0/456]	100% [456/456]
SOE	Wi-Fi	100% [352/352]	0% [0/352]

A mixed input to the classifiers (multi-network environment) was then considered. Given that the Wi-Fi capture is on real traffic, while the Bluetooth streams were simulated, the mixture could be controlled by software. In particular, three different mixs were generated: a) pre-dominant Wi-Fi (1000 Wi-Fi packets vs. 200 Bluetooth packets); b) balanced (1000 Wi-Fi packets vs. 1000 bluetooth packets); c) Bluetooth pre-dominant (1000 Wi-Fi vs. 2000 Bluetooth packets). All Bluetooth sequences were multi-slot. Wi-Fi captures were 1.4 seconds long, while Bluetooth simulations lasted 0.16, 0.75 and 1.6 seconds. Due to differences in the duration of captures only partial overlapping in the combined packet sequences was achieved. Results for this test are displayed on Table IV.

TABLE III
CLASSIFICATION RESULTS

	Classifier Input Network	Classification into Wi-Fi	Classification into multi-slot Bluetooth
Pocket	Bluetooth	0% [0/462]	100% [462/462]
Pocket	Wi-Fi	98.86% [348/352]	1.14% [4/352]
Perceptron	Bluetooth	0.43% [2/462]	99.57% [460/462]
Perceptron	Wi-Fi	98.86% [348/352]	1.14% [4/352]
LMS	Bluetooth	34.85% [161/462]	65.15% [301/462]
LMS	Wi-Fi	99.43% [350/352]	0.57% [2/352]
SOE	Bluetooth	29.87% [138/462]	70.13% [324/462]
SOE	Wi-Fi	99.72% [351/352]	0.28% [1/352]

TABLE IV
CLASSIFICATION RESULTS
MULTI-NETWORK ENVIRONMENT

Classifier	Input Network	Classification into Wi-Fi	Classification into multi-slot Bluetooth
Pocket	Bluetooth pre-dominant	17.10% [133/778]	82.90% [645/778]
Pocket	Wi-Fi pre-dominant	86.07% [315/366]	13.93% [51/366]
Pocket	Balanced	41.34% [210/508]	58.66% [298/508]
Perceptron	Bluetooth pre-dominant	17.22% [134/778]	82.78% [644/778]
Perceptron	Wi-Fi pre-dominant	86.07% [315/366]	13.93% [51/366]
Perceptron	Balanced	41.53% [211/508]	58.47% [297/508]
LMS	Bluetooth pre-dominant	37.79% [294/778]	62.21% [484/778]
LMS	Wi-Fi pre-dominant	90.16% [330/366]	9.84% [36/366]
LMS	Balanced	56.89% [289/508]	43.11% [219/508]
SOE	Bluetooth pre-dominant	36.89% [287/778]	63.11% [491/778]
SOE	Wi-Fi pre-dominant	90.71% [332/366]	9.29% [34/366]
SOE	Balanced	56.10% [285/508]	43.90% [223/508]

VI. DISCUSSION OF RESULTS AND FUTURE DIRECTIONS

As described in the above Section, network classification of Wi-Fi vs. Bluetooth was attempted based on the definition of two features: the maximum packet duration between two silence gaps, and duration of silence gaps. Four different classification algorithms were used: Pocket, Perceptron, LMS, and SOE

Results of classification showed that:

1) For the Wi-Fi vs. single-slot Bluetooth case (Table II), all proposed classifiers achieved perfect classification into the two classes, when one traffic stream (either Wi-Fi or Bluetooth) was given as input to the classifier. This result shows that the selected features were appropriate since they completely identify these two classes.

2) For the Wi-Fi vs. multi-slot Bluetooth case (Table III), classification is not as perfect as in the previous case, and depends upon classification algorithm as well as input data to the classifier. Among all the proposed classification strategies, Pocket and Perceptron emerge as the most successful and reliable, leading to a classification rate greater than 98%.

3) Data in Table IV speak to the adequacy of the classifiers in environments with heavy predominance of one technology, by their ability to reveal both technologies in each case. This ability is shown by comparing results of Pocket reported by Tables III and IV. As shown by tables, only Pocket and Perceptron are capable of performing a reliable classification. Note that these classifiers were always capable of providing as output, the pre-dominant network, and moreover, the rate of classification follows the trend in the proportion between both technologies packets in the observation sequence. When the traffic flows are balanced, the classifier seems to follow a "50-50" "win-win" rule, by outputting balanced classification decisions.

Future work will focus on investigating whether the selected features extend beyond the present case of two technologies in the ISM band. In particular, the AIR-AWARE project will proceed by incorporating the IEEE 802.15.4 technology (ZigBee) [12] into the set of possible classes. Preliminary investigations, based on the analysis of the 802.15.4 standard specifications, show that SIFS is also defined for ZigBee networks, with a nominal value of 192μs [12] in the ISM 2.4 *GHZ* band. This value compared to extracted features on this paper experiments, should allow the classification algorithms to obtain good separation for all three classes (Wi-Fi vs. Bluetooth vs. ZigBee).

REFERENCES

[1] IEEE Std 802.11 – 2007, IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 12 June 2007

[2] IEEE Std 802.15.1 – 2005, IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs), 14 June 2005.

[3] Theodoridis S. and Koutroumbas K., *Pattern recognition*, 4° Ed., Elsevier Inc., 2009.

[4] Schűrmann J., *Pattern classification – A unified view of statistical and neural approaches*, John Wiley & Sons Inc., 1996.

[5] Duda R.O., Hart P. E., and Stork D.G., *Pattern classification*, 2° Ed., Wiley-Interscience, 2004.

[6] Gallant S. I., Perceptron-Based Learning Algorithms, *IEEE Transactions on neural networks*, Vol. 1(2), 1990.

[7] http://netresearch.ics.uci.edu/kfujii/jpcap/doc/

[8] http://en.wikipedia.org/wiki/Monitor_mode

[9] http://www.radiotap.org/

[10] Ho Y.H. and Kashyap R.L. "An algorithm for linear inequalities and its applications," *IEEE Transactions on Electronic Computers*,Vol.14(5), 1965.

[11] Gandetto M. and Regazzoni C., "Spectrum Sensing: A Dsitributed Approach for Cognitive Terminals," *IEEE Journal on selected areas in communications*, Vol.25 (3), 2007.

[12] IEEE Std 802.15.4 – 2006, IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), 8 September 2006.

BIBLIOGRAPHY

[1] Wikipedia, Cognitive radio: http://en.wikipedia.org/wiki/Cognitive_radio

[2] Wikipedia, ISM band: http://en.wikipedia.org/wiki/ISM_band

[3] Sikora A., and Groza V.F., "Coexistence of IEEE 802.15.4 with other Systems in the 2.4 GHz-ISM-Band", *IMTC 2005 – Instrumentation and Measurement Technology Conference*, Ottawa, Canada, 17-19 May 2005

[4] IEEE Std 802.15.1 – 2005, IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs), 14 June 2005

[5] Bluetooth SIG, *Specification of the Bluetooth system*, 17 December 2009

[6] Lindholm T., "Setting up a Bluetooth Packet Transport Link", Department of Computer Science, Helsinki University of Technology

[7] Chang D.-C., "Least Squares/Maximum Likelihood Methods for the Decision-Aided GFSK Receiver", *IEEE Signal Processing letters*, Vol. 16 (6), June 2009

[8] Svedek T., Herceg M., and Matic T., "A Simple Signal Shaper for GMSK/GFSK and MSK Modulator Based on Sigma-Delta Look-up Table", *Radioengineering*, Vol. 18 (2), June 2009

[9] Kontakos N.P., and Pollard J.K., "Bluetooth RF Layer Performance Evaluation", University College London

[10] Benedetto S., Biglieri E., and Castellani V., *Digital transmission theory*, Prentice-Hall Inc., 1987

[11] Lee E.A., and Messerschmitt D.G., *Digital communication*, 2nd ed., Kluwer Academic Publishers, 1994

[12] Biglieri E., Divsalar D., McLane P.J., and Simon M.K., *Introduction to trellis-coded modulation with applications*, Macmillan Publishing Company, 1991

[13] Anderson J.B., and Mohan S., *Source and channel coding – an algorithmic approach*, Kluwer Academic Publishers, 1991

[14] Mikéska Z., and Hanus S., "The inquiry procedure in the Bluetooth networks", Institute of Radio Electronics, FEEC, Brno University of Technology

[15] Schiphorst R., Hoeksema F., and Slump K., "Channel selection requirements for Bluetooth receivers using a simple demodulation algorithm", Laboratory of Signals and Systems, Department of Electrical Engineering, University of Twente

[16] Li J., Liu X., and Ma X., "Dual Channel Transmission for Coexistence of Bluetooth Piconets with Multi-Slot Packets", *IEEE*, 2007

[17] Jesus Roldan Diaz, "Cognitive networking: network sensing with application to IEEE 802.11 communication systems", MSci degree in Telecommunication Engineering, Sapienza Università di Roma, 12 April 2010

[18] Carmen Juana Martin Martin, "Towards cognitive networking: automatic recognition of technologies operating in the ISM bands", MSci degree in Telecommunication Engineering, Sapienza Università di Roma, 12 April 2010

[19] Spill D., "Final Report: Implementation of the Bluetooth stack for software defined radio, with a view to sniffing and injecting packets", MSci degree in

Computer Science, University College London, 14 May 2007

[20] Java APIs for Bluetooth Wireless Technology (JSR 82), Specification Version 1.1.1, Java 2 Platform, Micro Edition, Sun Microsystems Inc., 29 July 2008

[21] Spill D., and Bittau A., "BlueSniff: Eve meets Alice and Bluetooth", University College London

[22] Moser M., "Busting The Bluetooth Myth – Getting RAW Access – aka Transforming a consumer Bluetooth Dongle into a Bluetooth Sniffer", http://www.remote-exploit.org

[23] Karacho K., "Do It Yourself Bluetooth Sniffer", 16 May 2007

[24] Laurie A., Holtmann M., and Herfurt M., "Hacking Bluetooth enabled mobile phones and beyond – Full Disclosure", December 2004