SAPIENZA
UNIVERSITÀ DI ROMA

Facoltà di Ingegneria

Tesi di Laurea Specialistica in:

INGEGNERIA DELLE TELECOMUNICAZIONI

TOWARDS COGNITIVE NETWORKING:

Automatic Recognition of Technologies Operating in the ISM Bands

Candidata:

Carmen J. Martin Martin

Relatore:

Prof.ssa Maria-Gabriella Di Benedetto

Anno Accademico

2009/2010

Acknowledgment

I've always thought that a man is just as big as the size of his efforts... However, many times whatever we achieve depends not only on us, but on the wonderful people God has put right beside us. Today, I'd like to thank some of that people:

First off, my Family –the greatest blessing God has provided me with. A set of parents that have been the best role models for perseverance, everlasting love, honesty and humbleness. Thank you for supporting me all the way and believing in me even at those times when I wouldn't. To Veronica, for being the best sister in the whole wide world: you always cheered me up and you bear with my nighttime bad moods. I love you, my Vero! To my three little ones –Vicky, Isa and Gaby – I love you like crazy; you fill my life with joy, light and love. To my aunts and uncles, grandparents and godparents for trusting me and being there for me in the hardest of times. All of you are my cherished Treasure.

To my girl-friends, for being there and teaching me the true meaning of Friendship: someone that's always there in the happiest moments and the not-so-happy ones, someone that knows no distance... I love you guys.

To a very special being, an angel –wherever you are I know you'd be happy to see me becoming a full-fledged Engineer.

To my "Gordo", even when I know I barely ever say it: Thanks! For being alive, for bringing joy to my life and for being an awesome dissertation partner. I admire you. I am very proud of you.

To Cris, for adopting me as a little sister, for putting up with the sleepless nights and nuisances, without you this road would have been fair wearier... Thanks, pequegna!

To the people that in the last few months has become our Family, making us feel at home: To Prof.ssa Maria-Gabriella for being the paragon of intelligence, beauty and warmth of heart. To Luca and Dome for being patient and giving us the opportunity to learn something new day in and day out. To Stefano and Sergio for offering us their Friendship,

company and for giving us a good laugh in the most stressful moments at the Lab. You are all very special to me!

Last but not least, to God for giving a new chance at life, laughter and dreaming every single day.

I am incredibly grateful to all of you.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Cognitive Radio

Cognitive means related to knowledge – accumulated information derived from experience or learned through introspection. Cognitive development focuses in studying thinking processes and the behavior derived of those. In this sense, "cognitive" is a fitting adjective to describe the paradigm of wireless communications, where both networks and nodes change specific parameters of transmission and reception to adjust their functioning to a mechanism of observation and learning from environmental factors –such as, radiofrequency spectrum, user behavior and network status.

Over the last few years, many research studies shown that spectrum use is related to timing and location. Specific assignation of the spectrum poses a problem: frequencies assigned to services with low use, are not available to unauthorized users, even when these transmission will not cause any interference in this un-occupied service. This used to be the reason why unauthorized users made use of bands that required authorization: they assumed their work will cause no interference, since whenever a legitimate user needed to make a transmission, they could jump to another frequency-band to continue their transmissions. Cognitve radio was designed with the intention of enabling users to seize these temporary voids in the electromagnetic spectrum.

Depending on the environment and set of parameters considered to make a decision about alterations to transmission and reception, we can distinguish some types of cognitive radio. The term "full cognitive radio" (or Mitola's Radio) refers to that in which any observed parameter in a wireless node or network is taken into account for decision-making. On the other hand, "spectrum detecting cognitive radio" observes only radiofrequency spectrum status and makes a decision based on this parameter; depending on spectrum's availability, notice the following subtypes:

- ✓ Licensed bands: when cognitive radio can use bands assigned to licensed users, asides from free access bands like ISM or UNII[1].
- ✓ Unlicensed bands: when only free access parts of the radio frequency spectrum can be used.

The most relevant requisite for cognitive radio functioning is their ability to detect unused spectrum and used it without provoking negative interference for other users. The best way to find these "usage voids" is detecting legitimate authorized users. Detection techniques can be categorized as follows:

- *Transmission Detection*: cognitive radios need to possess the ability to determine if there is signal from any users accessing a part of the spectrum.
- *Cooperative Detection*: different cognitive radio users periodically exchange information about detection of principal users.
- *Interference-based detection:* Another relevant function is spectrum administration –using bandwidth in a way that fits best the QoS required by the user, and selecting this bandwidth amongst those available. There are two steps to spectrum administration: spectrum analysis and spectrum decision. The first one refers to identifying the characteristics of each available band, assessing for advantages or obstacles (such as delay and error probability). The second step compares characteristics between bands with user's needs and evaluates which one is will make the best fit.

[1] Unlicensed National Information Infrastructure

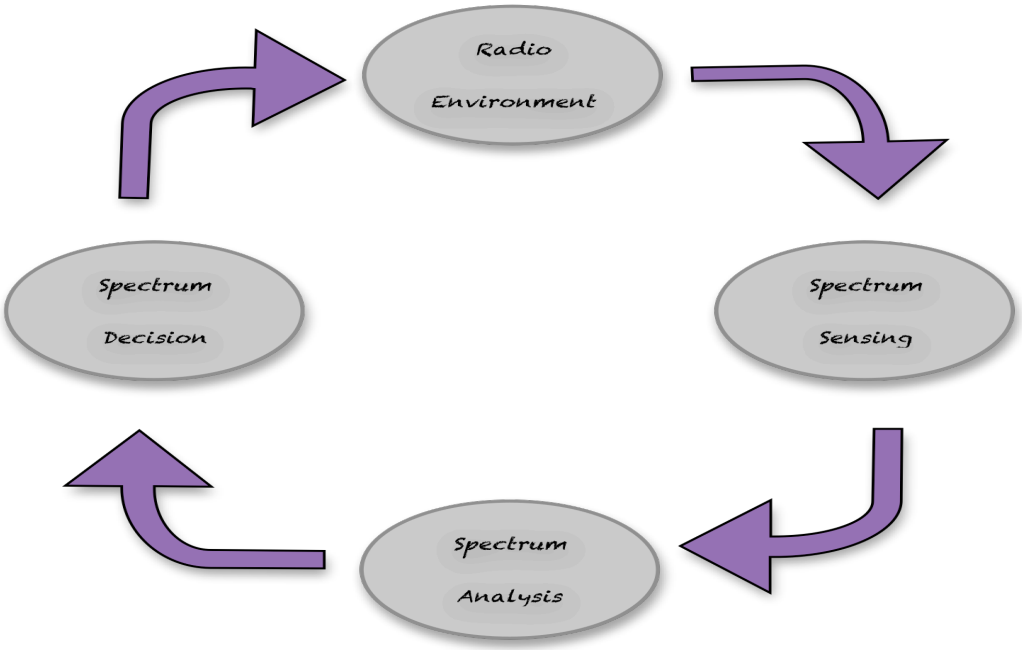


Figure 1.1 - Cognitive Radio operation

A fundamental advantage of cognitive radio is spectral mobility, the process by which a cognitive radio changes its frequency of transmission or reception. Cognitive radios are designed to change bands constantly, choosing the best available options in a way that is imperceptible for users. Finally, another defining aspect of cognitive radios is the ability to share the spectrum. This is achieved by a schematic method of spectrum's distribution that is fair and egalitarian for every cognitive radio user without interfering with authorized user's transmissions. This poses one of the greatest challenges of designing a cognitive radio, as do generic issues of media access we face nowadays.

1.2 Unlicensed Bands

The ISM (Industrial Scientific and Medical) bands were defined by ITU-R[2]. Three sub-bands composed the ISMs:

- 902-928 MHz
- 2400-2483.5 MHz
- 5725-5850 MHz

[2] International Telecommunication Union- Radiocommunication Sector

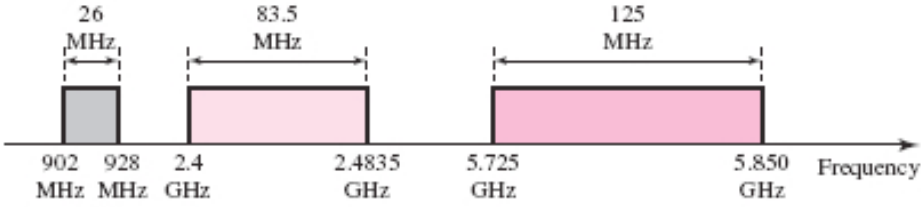


Figure 1.2 - Industrial Scientific and Medical (ISM) bands

In the beginning, ISM radio bands were assigned uniquely to industrial, medical and scientific endeavors. In 1985, the Federal Communications Commission (FCC) issued rules permitting "intentional radiators" to use these bands, however some restrictions were outlined:

1. Maximum transmitter output is 1W (30 dBm) .
2. Maximum EIRP[3] is 4W (36 dBm) .
3. For fixed point to point operation in ISM2.4, peak output need only be reduced by 1 dBm for every 3 dBi of antenna gain above 6 .
4. In ISM5.8, you can apply all the antenna gain you want without reduction in output power.

The next table shows the common devices operating in the ISM bands:

902-928 MHz	*2.4-2.4835 GHz*	*5.725-5.85 GHz*
• *Cordless Phones* • *Cordless Headphones* • *Surveillance systems*	• *IEEE LAN Standards.* • *Audio/video signal repeaters* • *Remote garage openers* • *Microwave oven*	• *Reserved for high bit rate networking devices* • *IEEE/ETSI LAN Standards*

Table 1.1 - Devices operating in the ISM bands

[3] Equivalent Isotropically Radiated Power

Interest in using these bands has been stimulated by several factors that differ substantially for the European approach of conscientious, but time consuming standardization. Most importantly, there is almost a complete absence of user restrictions (no registration procedure, no qualification of end users) as to where the products can be used. The absence of license fees also contributes to financial attractiveness of products.

1.3 Background and Motivations

As part of the incessant pursuit to guarantee completely aware wireless communication behavior, automatic network recognition has become a promising attribute dedicated to integrate cognitive mechanisms over the network layer, enabling us to come closer to our main objective: Cognitive Networking .

Certainly, as detailed in Section 1.1, spectrum sensing plays a key role in a Cognitive Radio, but in order to provide a qualitative description of the spectrum, air interfaces classification is also performed [1].

This work proposes an automatic recognition approach base on extraction of features that best reveal MAC sublayer communication procedures. That means that in order to recognize the different technologies operating in the ISM band, an analysis was performed at the level of the data link layer, with the goal of finding features with sufficient discriminatory power so that optimal classification criterion that will report good results –in a simple and low cost fashion.

This approach include two fundamental phases: choice and extraction of features, and implementation of a linear classification algorithm that decides which technologies are in the air –reporting a percentage.

Work was focused in Wi-Fi versus Bluetooth recognition. In this case, four different algorithms were implemented and tested. This stage of evaluation will demonstrate the validity of the chosen features, as well as the performance of each of the classification algorithms utilized.

1.4 Organization of the work

In Chapter 2, an exhaustive study about WLAN (IEEE802.11) was developed with the objective of analyzing fundamental characteristics that reflect communications procedures of the MAC sublayer, so that possible features to use on the algorithm could be identified. The study of this technology afforded us a better understanding and knowledge of relevant aspects and directing this work to a certain destination.

On the other hand, Chapter 3 exposes a complete description of pattern recognition, as well as linear classification algorithms that are commonly utilized. In Chapter 4 exhaustive definition of the issue of automatic recognition, with details about generation of training sets and implementation of the classification block. Later on, the experimentation section is presented with a further analysis stage.

Finally, conclusions and directions are presented to expose the results of the chosen approach and pointing out to new and interesting research journeys.

Chapter 2

WIRELESS LAN MEDIUM ACCESS CONTROL (MAC) AND PHYSICAL LAYER (PHY) SPECIFICATIONS

2.1 Overview

A Wireless LAN [2] (Local Area Network - WLAN) is a data communication system that can be use as an alternative to wired LANs or as an extension to an existing one. Coverage usually extends between 10 a 100 meters; this limited reach affords lower transmission power that allows the use of unlicensed bands. WLANs operate under radiofrequency technology, which makes them ideal for greater flexibility, mobility, immediate access for temporal users and ease of installation.

The IEEE (Institute of Electronics and Electrical Engineering) by way of one of their standards (the 802.11 specifically) regulates functioning of WLANs. Standard specifications are guided by the OSI (Open System Interconnection) model, that focus in its inferior levels (Physical Layer and MAC Sub-layer).

This standard [3] is part of a list enforceable for local and metropolitan area networks. The relationship between this and other standards can be illustrated as follows.

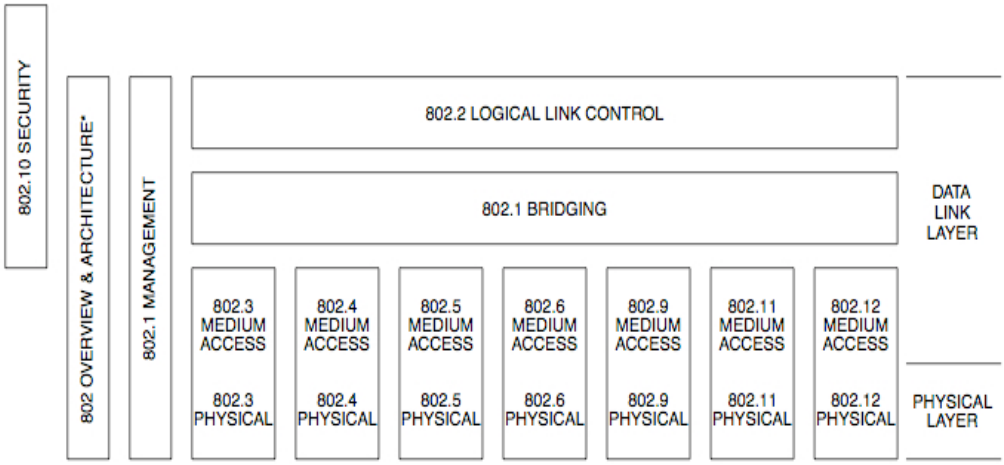


Figure 2.1 - IEEE 802 Family

The ANSI/IEEE 802.11, 1999 Edition, is a version of the original IEEE 802.11 standard, published in 1997, with modifications such as deletion of redundant management item and completion of one of the annexes.

The ANSI/IEEE 802.11, 1999 Edition defines protocol and compatible interaction of data communications equipment via the air, radio or infrared, on a local area network (LAN) using the carrier sense multiple access protocol with collision avoidance (CSMA/CA) medium sharing mechanism. The medium access control (MAC) supports operations, either under access point control or between independent stations. The protocol includes authentication, association, and re-association services, optional encryption/decryption procedure, power management and a point coordination function for time-bounded transfer data.

Much in the same way that the MAC sub-layer establishes rules to determine medium access and data transmission, details of transmission and reception are regulated by the Physical Layer (PHY).

The original version of the IEEE 802.11 standard became obsolete and was fine tuned in 1999. Back then, it specified two net bit rates of 1 or 2 megabits per second (Mbit/s), plus forward error correction code and only three alternative physical layer technologies: diffuse infrared operating at 1 Mbit/s; frequency-hopping spread spectrum operating at 1 Mbit/s or 2 Mbit/s; and direct-sequence spread spectrum operating at 1 Mbit/s or 2 Mbit/s. The latter two radio technologies used microwave transmission over the Industrial Scientific Medical frequency band at 2.4 GHz. Some earlier WLAN technologies used lower

frequencies, such as the U.S. 900 MHz ISM band.

In 2003, task group TGma was authorized to "roll up" many of the amendments to the 1999 version of the 802.11 standard. REVma or 802.11ma, as it was called, created a single document that merged 8 amendments (802.11a,b,d,e,g,h,i,j) with the base standard. Upon approval on March 8, 2007, 802.11REVma was renamed to the current base standard IEEE 802.11-2007.

Currently, different types of PHYs are in use under 802.11, with a broad range of versions that can be categorized by utilized band, modulation type and coding; this characteristics determine distinct transmission rates and throughputs. A brief description of the available versions is provided here [2]:

802.11 a

The 802.11a standard uses the same data link layer protocol and frame format as the original standard, but an OFDM based air interface (physical layer). It operates in the 5 GHz band with a maximum net data rate of 54 Mbit/s.

Since the 2.4 GHz band is heavily used to the point of being crowded, using the relatively un-used 5 GHz band gives 802.11a a significant advantage. However, this high carrier frequency also brings a disadvantage: the effective overall range of 802.11a is less than that of 802.11b/g. In theory, 802.11a signals are absorbed more readily by walls and other solid objects in their path due to their smaller wavelength and, as a result, cannot penetrate as far as those of 802.11b. In practice, 802.11b typically has a higher range at low speeds (802.11b will reduce speed to 5 Mbit/s or even 1 Mbit/s at low signal strengths). However, at higher speeds, 802.11a often has the same or greater range due to less interference.

802.11b

802.11b has a maximum raw data rate of 11 Mbit/s and uses the same media access method defined in the original standard. 802.11b products appeared on the market in early 2000, since 802.11b is a direct extension of the modulation technique defined in the original standard. The dramatic increase in throughput of 802.11b (compared to the original standard) along with simultaneous substantial price reductions led to the rapid acceptance of 802.11b as the definitive wireless LAN technology.

While using a spread spectrum technique based on DSSS, the 802.11b extension introduces CCK (Complementary Code Keying) to achieve rates of 5,5 and y 11 Mbps . This standard also supports PBCC (Packet Binary Convolutional Coding) as an optional. All 802.11b devices must maintain compatibility with prior DSSS equipment, as specified in the original IEEE 802.11 regulation, with rates of 1 and 2 Mbps.

802.11c

The "c" protocol is used for communication between different networks or different types through a wireless connection, as well as connection between distant buildings. The 802.11c is a modified version of 802.1d that offers no advantages for the general public; but allows to combine 802.1d with devices that comply with 802.11 (on the Data Link Layer). While of less common use than its two predecessors, this protocol offers advantages –in time and budget- over optic fiber installations to establish larger distance communications.

802.11d

A supplement to 802.11, this standard is designed to support international use over 802.11 local networks. The idea is to allow dispositive to interchange information using the frequency range permitted by their country of origin.

802.11e

This standard offers real time applications, through its Quality of Service warranty. 802.11e is designed to support real time traffic regardless of the environment or situation. The objective with this standard was to introduce new mechanisms on the MAC layer in order to support services that will require QoS. To achieve this, Hybrid Coordination Function (HCF) with two types of access was introduced:

- (EDCA) Enhanced Distributed Channel Access, equivalent a

DCF.

- (HCCA) HCF Controlled Access, equivalent to PCF.

In this new standard four access categories are defined (in descending priority order):

- Background (AC_BK)
- Best Effort (AC_BE)
- Video (AC_VI)
- Voice (AC_VO)

To achieve traffic differentiation medium access times and contention window sizes are defined for each category.

802.11f

Recommended for access point providers, it improves access point compatibility. Utilizes the IAPP[4] protocol to achieve optimal itinerancy: allowing traveling users to change between access points while on the move, regardless of the brand of the access points on network infrastructure.

802.11g

Made public in June 2003, 802.11g is an evolution of the b standard that uses the 2.4 GHz band with a maximum theoretical rate of 54 Mbps. It is compatible with the 802.11b and uses the same frequencies. Currently in the market, there are "g" standard devices with up to a half watt power, which will support communications in a range of 50 kilometers, provided the use of parabolic antennas or appropriate radio equipment.

802.11h

The 802.11h specification for WLANs, developed by Workgroup 11 of the IEEE LAN/MAN Committee (IEEE 802) and made public in October 2003, was designed to troubleshoot issues derived from coexistence of 802.11 networks and Satellite / Radar systems.

This development follows ITU recommendations made after the ERO (European

[4] Inter –Access Point Protocol

Radiocommunications Office) requirements to minimize the impact of opening the 5 GHz band to ISM applications.The 802.11h incorporates the capability to manage dynamically both frequency and power of transmissions (Dynamic Frequency Selection and Transmitter Power Control). DFS allows WLANs operating on 5 GHz band to avoid co-channel interference with Radar systems and ensures uniform utilization of available channels; while TPC enforces required potency limitations for each regional channel, avoiding interference with satellite systems.

802.11i

Developed to counter the current vulnerability of authentication and coding protocols, encompasses the following protocols: 802.1x, TKIP (Temporal Key Integrity Protocol) and AES (Advanced Encryption Standard). The 802.11i is implemented on WPA2.

802.11j

An equivalent of 802.11h, designed to comply with Japanese regulations.

802.11k

Allows commuters and wireless access points to value and calculate the radiofrequency resources of a WLANs clients, improving its management. Design to be implemented as software, LAN equipment is able to support it after updates. For the standard to be effective, clients (WLAN adapters and cards) and infrastructure (access points and commuters) must be compatible.

802.11n

In January 2004, IEEE announced the formation of workgroup 802. 11 to develop a revision of the standard. Transmission rate could reach 600 Mbps and it will be up ten times faster than 802.11a and 802.11g, and at times faster than 802.11b. Also, improvements in range are expected thanks to the MIMO Multiple Input – Multiple Output technology, which integrates several antennas to allow use of multiple channels during transmission. A series of delays has plagued the development with a new deadline on November 2009. Unlike other versions, 802.11n can work on double frequency bands (2.4 GHz and 5 GHz). This capability makes the new standard compatible with all previous versions.

802.11p

This standard operates on the 5.9 GHz frequency band, recommended for

automobiles it will be the building block of short range communications in the United States. DSRC[5] technology will allow data interchange between vehicles and road infrastructure.

802.11r

Also known as Fast Basic Service Set Transition, its most relevant characteristic is allowing the network to establish security protocols that identify a device in a new Access Point before abandoning the previous one. This function provides a transition time of less than 50 milliseconds. In a VoIP communication, there will be no perceptible interruptions.

802.11s

Defines manufacturer interoperability regarding Mesh protocols (networks that combine two topologies: ad-hoc and infrastructure). Because there is no established standard, each manufacturer has its own mesh generation mechanism.

802.11v

IEEE 802.11v (coming in 2010) will allow remote configuration of client devices, providing in turn centralized (similar to a cellular network) or distributed (through a layer 2 mechanisms) station management. This includes network ability to supervise, configure and update client stations. Besides management improvement, new capabilities of the 11v are: energy savings for PDAs, positioning for services that depend on location, temporization to support fine caliber applications and coexistence between different technologies within the same device.

802.11w

Still in development, it's being designed to improve the layer of medium access control, to increase security on authentication and coding protocols. Currently, WLANs send system information in unprotected frames that makes the network vulnerable. This standard is created to protect networks against interruptions caused malware that creates fake request from unassociated stations that look like they were sent from a valid device. The 11v attempts to extend the protection from the data to the management frames to ensure security on the network's vital operations.

---

[5] Dedicated short-range communications

802.11y

Published on November 2008, allows operation on 3650 to 3700 MHz bands (except when interference can be generated) in the United States. Three new concepts are wrapped around this standard: Contention Base Protocol (CBP), Extended Channel Switch Announcement (ECSA), and Dependent Station Enablement (DSE). CBP includes improvements in the detection mechanisms of portability. ECSA provides a mechanism for the APs to notify their stations its intention to switch channels or bandwidths. Lastly, DSE is used to manage licenses.

2.2 General Description of the architecture

According to the IEEE 802.11 standard [3], the WLAN's architecture consists in various components interacting to create a WLAN that supports station mobility transparently to upper layers.

The Basic Service Set (BSS) is the building block of an IEEE 802.11 LAN. The following figure displays two BSSs, each one with two client stations.

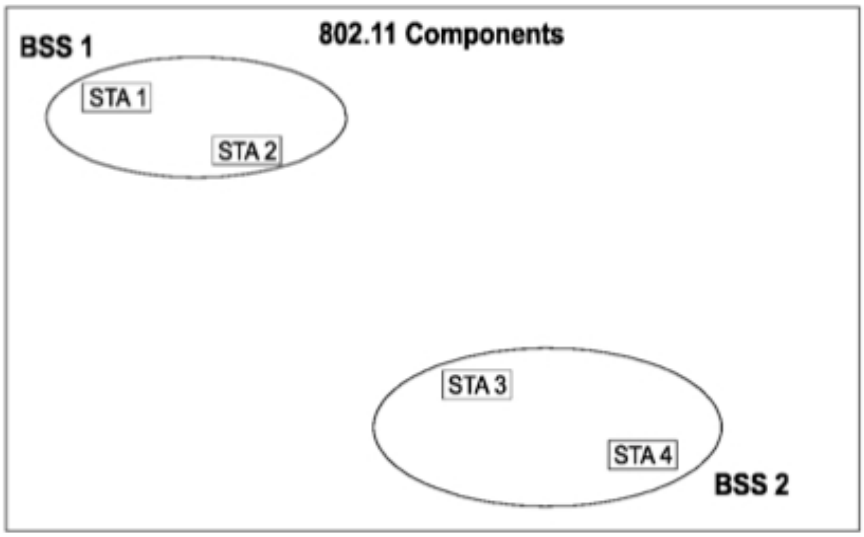


Figure 2.2 - BSS Structure

The ovals describe BSSs coverage areas through which member stations can remain communicated. If a member station moves out of range, it will no longer be able to establish communication with other BSS's members.

2.2.1 The independent BSS as an ad hoc network

The independent BSS (IBSS) is the most basic IEEE 802.11 LAN type. The smallest IEEE 802.11 LAN could consist of two stations. Figure 2.2 shows two IBSSs. This modality of IEEE 802.11 operation is possible when stations are able to communicate directly. Because this type of LAN is many times built without pre-planning, this implementation is often referred to as an "ad hoc network".

2.2.2 Distribution System Concepts

A BSS can also be part of a larger network built with multiple BSSs. The architectural component utilized to interconnect several BSSs is the Distribution System (DS). IEEE 802.11 separates logically the wireless medium (WM) from the Distribution System Medium (DSM). The IEEE 802.11 LAN architecture is independent of the physical characteristics and implementation specifications. The DS allows mobility to the devices, purveying the necessary logical services to manage address to destination mapping and seamless integration of multiple BSSs.

The Access Point (AP) is the station (STA) that gives access to the DS, offering more functionalities than a simple station.

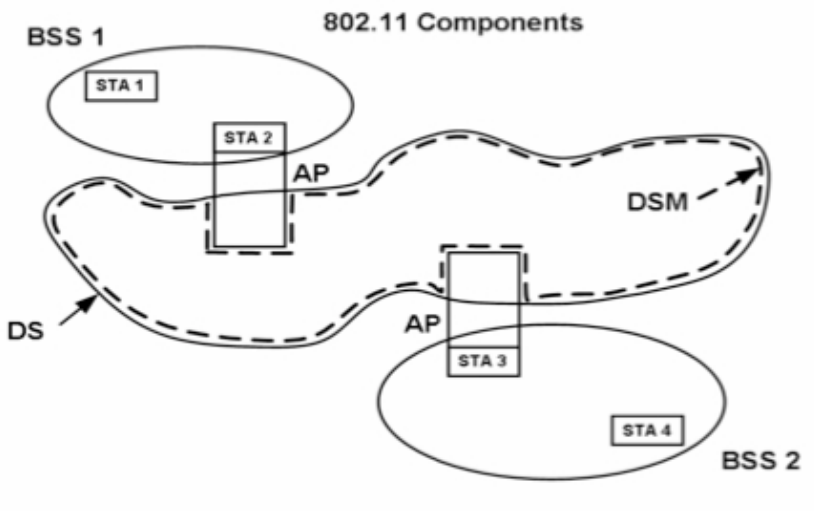


Figure 2.3 - DS and AP

The data moves between BSS and Ds, through the AP. Every AP is also and station and as such, an addressable entity. Both DS and BSSs allow the IEEE 802.11 to create a wireless network of arbitrary size and complexity, or an Extended Service Set Network.

The key concept here is that an ESS network is at the same LLC (Logical Link Control) Layer that IBSS network. The interior stations of an ESS can communicate, as well as the mobile stations can move from a BSS to another internally on the ESS- while being transparent to the LLC.

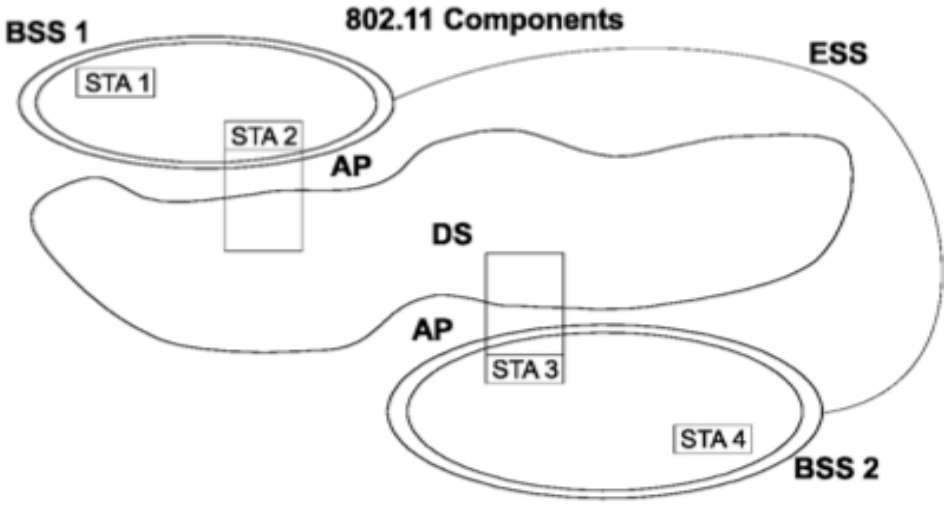


Figure 2.4 - ESS Structure

In this setting, any of these are possible:

- BSSs could partially overlap.
- BSSs could be physically separated.

One or more IBSS or ESS networks could be present in the very same physical space.

2.2.3 Integration with wired LANs

To integrate IEEE 802.11 architecture with a traditional wired LAN, a logical component needs to be introduced: a portal. The portal is a logical point through which MSDUs originated in a non-IEEE 802.11 LAN enter the IEEE 802.11 DS. As displayed on the figure 2.5, all data coming from the 802.xLAN enters the IEEE 802 architecture through the portal. The portal provides logical integration between IEEE 802.11 architecture and the existing wired LANs. It is possible that a device is designated as both an AP and a portal.

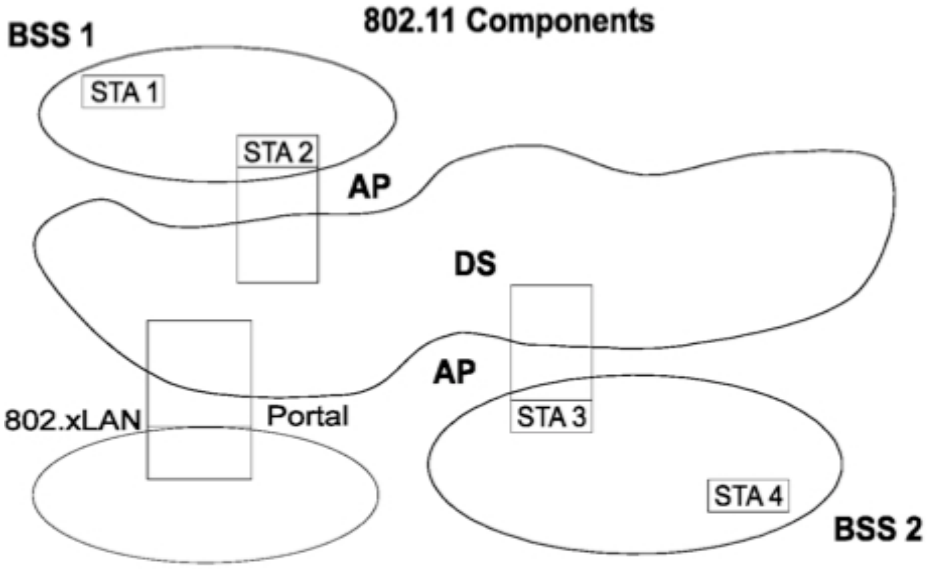


Figure 2.5 - Connecting to other IEEE 802 LANs

2.2.4 Logical Service Interfaces

IEEE 802.11 doesn't explicitly specify the details of DS implementation. Instead, it specifies the services in two categories: the station service (SS) and the distribution system service (DSS). Both categories are utilized by the MAC sub-layer.

Services provided by stations (SS) are:

a. Authentication
b. Deauthentication
c. Privacy
d. MSDU delivery
e. DFS
f. TPC[6]
g. Higher layer timer synchronization (QoS facility only)
h. QoS traffic scheduling (QoS facility only)

The following are Distribution System Services (DSS):

a. Association
b. Disassociation
c. Distribution
d. Integration
e. Reassociation
f. QoS traffic scheduling (QoS facility only)

[6] Transmit Power Control

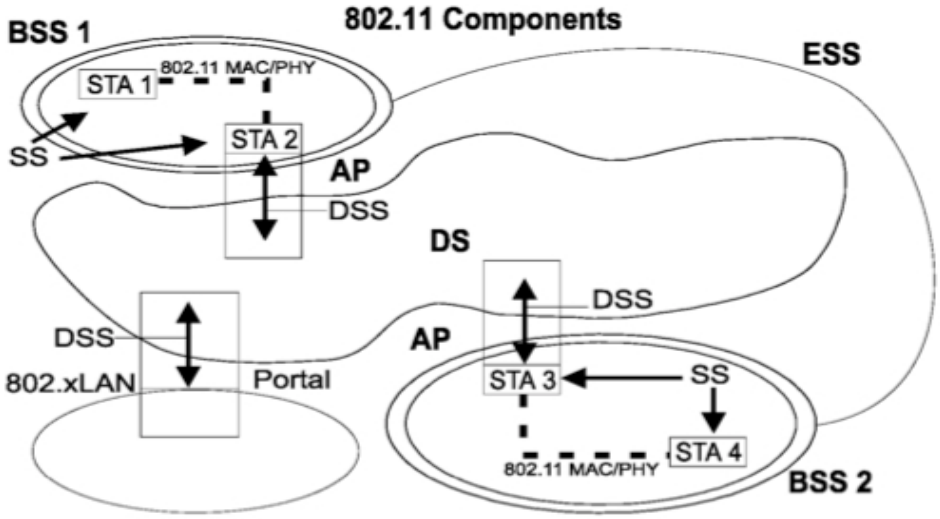


Figure 2.6 - Complete IEEE 802.11 architecture.

As Figure 2.6 illustrates, DSS are represented by arrows towards the APs, labeling these services as used to cross media and address space logical boundaries; while SS are represented by arrows towards stations (STAs).

2.3 Frame Structure

A MAC frame structure is composed by a MAC Header, a variable length frame body and a frame check sequence (FCS). Figure 2.7 displays a frame's general structure:

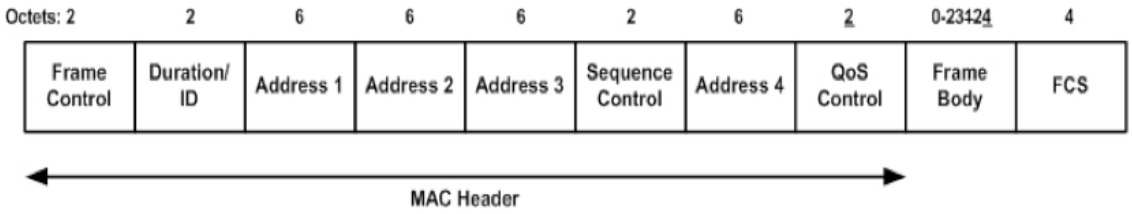


Figure 2.7 - Frame format

There is sufficient information on the MAC header to manage fragmentation, transmission, encryption and data being transported on the packet. The length of the header, as well as the data, is variable and depends on the type of frame that is being transmitted. For

this reason, the fields Address 2, 3 and 4, and Sequence Control may or may not be present. A 32-bit Cyclic Redundancy Code (CRC) is stored in the FCS to verify the frame's integrity.

Here's a detailed explanation of each field that integrates the MAC Frame:

2.3.1 Frame Control

This field is composed of two bytes integrating the fields Protocol Version, Type, Subtype, To DS, From DS, More Fragments, Retry, Power Management, More Data, Protected Frame and Order. Figure 2.8 illustrates Frame Control structure.

B0 B1	B2 B3	B4 B7	B8	B9	B10	B11	B12	B13	B14	B15
Protocol Version	Type	Subtype	To DS	From DS	More Frag	Retry	Pwr Mgt	More Data	Protected Frame	Order
Bits : 2	2	4	1	1	1	1	1	1	1	1

Figure 2.8 - Frame Control field

The Protocol Version field, 2 bits in length, specifies the frame's version of 802.11 MAC. Up until now, there is only one version and therefore the field value is 0; in the future, we might find further versions and different values.

Type and Sub Type fields identify the type of frame that is being used. Values on this field depend on the type of data being transmitted. Three types of frame are available:

- Control
- Management
- Data

Each type contains subtypes. In the data frame ,the most relevant bit on the subtype field, indicates is QoS functionalities are supported. This bit has come to be known as the QoS subfield.

Frame types with their subtypes are displayed on tables 2.1 and 2.2.

Type value b3 b2	Type description	Subtype value b7 b6 b5 b4	Subtype description
00	Management	0000	Association request
00	Management	0001	Association response
00	Management	0010	Reassociation request
00	Management	0011	Reassociation response
00	Management	0100	Probe request
00	Management	0101	Probe response
00	Management	0110–0111	Reserved
00	Management	1000	Beacon
00	Management	1001	ATIM
00	Management	1010	Disassociation
00	Management	1011	Authentication
00	Management	1100	Deauthentication
00	Management	1101	Action
00	Management	1110–1111	Reserved
01	Control	0000–0111	Reserved
01	Control	1000	Block Ack Request (BlockAckReq)
01	Control	1001	Block Ack (BlockAck)
01	Control	1010	PS-Poll
01	Control	1011	RTS
01	Control	1100	CTS
01	Control	1101	ACK
01	Control	1110	CF-End
01	Control	1111	CF-End + CF-Ack

Table 2.1 - Frames Types and Subtypes (1)

Type value b3 b2	Type description	Subtype value b7 b6 b5 b4	Subtype description
10	Data	0000	Data
10	Data	0001	Data + CF-Ack
10	Data	0010	Data + CF-Poll
10	Data	0011	Data + CF-Ack + CF-Poll
10	Data	0100	Null (no data)
10	Data	0101	CF-Ack (no data)
10	Data	0110	CF-Poll (no data)
10	Data	0111	CF-Ack + CF-Poll (no data)
10	Data	1000	QoS Data
10	Data	1001	QoS Data + CF-Ack
10	Data	1010	QoS Data + CF-Poll
10	Data	1011	QoS Data + CF-Ack + CF-Poll
10	Data	1100	QoS Null (no data)
10	Data	1101	Reserved
10	Data	1110	QoS CF-Poll (no data)
10	Data	1111	QoS CF-Ack + CF-Poll (no data)
11	Reserved	0000–1111	Reserved

Table 2.2 - Frames Type and Subtypes (2)

The Fields *To DS and From DS* indicates whether the packet is addressed to the DS or not. The values on these fields also help determine which addresses must be situated in the fields Address 1, 2, 3 and 4. Table 2.3 offers an interpretation of DS field's value.

To DS and From DS values	Meaning
To DS = 0 From DS = 0	A data frame direct from one STA to another STA within the same IBSS, or a data frame direct from one non-AP STA to another non-AP STA within the same BSS, as well as all management and control frames.
To DS = 1 From DS = 0	A data frame destined for the DS or being sent by a STA associated with an AP to the Port Access Entity in that AP.
To DS = 0 From DS = 1	A data frame exiting the DS or being sent by the Port Access Entity in an AP.
To DS = 1 From DS = 1	A data frame using the four-address format. This standard does not define procedures for using this combination of field values.

Table 2.3 To DS and From DS subfields

The field *More Fragments* was designed to manage frame fragmentation. Field's value equals 1 when subsequent fragments of a data or management frame are expected. Otherwise, More Fragments will equal 0.

The *Retry* field helps the destination device to avoid processing duplicated frames. When Retry equals 1, the frame is a retransmission.

Since the 802.11 standard is oriented to mobile devices such as laptops and PDAs, the *Power Management* field provides support for the energy consumption of these devices. When Power Management equals 1 the transmitting device can turn to power save mode; when the value is 0, the station is in active mode. In Access Points this value is always 0, since these devise don't possess energy support capabilities. However, Access Points have the ability to store packets destined to "sleeping" devices.

The *More Data* field informs "sleeping" devices that they have packets awaiting reception (More Data = 1); to retrieve these, devices must send PS-Poll packets to their APs.

The *Protected Frame* field manages confidentiality and data authentication; when its value is 1, it indicates that encryption has been applied to the packet and the structure is slightly changed. The fixed value of PF is 1 for Data Frames as well as Authentication subtype.

The security of a wireless LAN is very important, especially for applications hosting

valuable information. For example, networks transmitting credit card numbers for verification or storing sensitive information are definitely candidates for emphasizing security. In these cases and others, proactively safeguard your network against security attacks.

This standard defines two classes of security algorithms for IEEE 802.11 networks [3]:

- Algorithms for creating and using an RSNA [7], called *RSNA algorithms*
- Pre-RSNA algorithms

Pre-RSNA algorithms includes the WEP-40. This was defined as a means of protecting (using a 40-bit key) the confidentiality of data exchanged among authorized users of a WLAN from casual eavesdropping. The figure 2.9 depicts the basic WEP encryptions : RC4 keystream XORed with plaintext Standard 64-bit WEP uses a 40 bit key (also known as WEP-40), which is concatenated with a 24-bit initialization vector (IV) to form the RC4 traffic key.

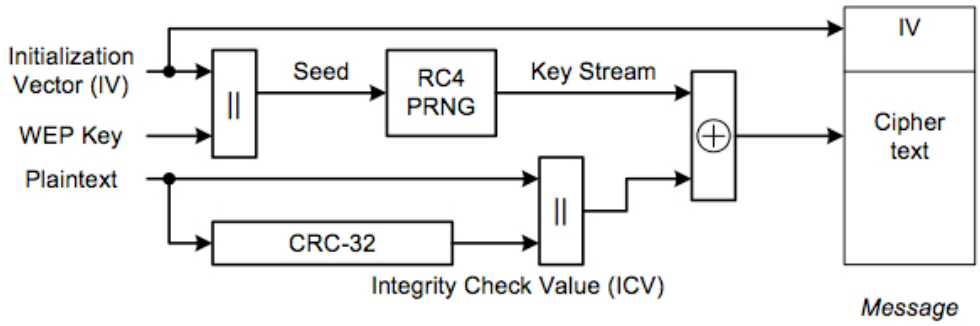


Figure 2.9 - WEP procedure

Sometimes packet order is relevant to the transmission and processing costs, the *Order* field takes care of this function. When its value equals one, strict packet delivery order must be followed.

2.3.2 Duration/ID

This 16 bit field can be used one of three ways. Table 2.4 illustrates the possibilities.

[7] Robust Security Network Association

Bits 0–13	Bit 14	Bit 15	Usage
0–32 767		0	Duration value (in microseconds) within all frames other than PS-Poll frames transmitted during the CP, and under HCF for frames transmitted during the CFP
0	0	1	Fixed value under point coordination function (PCF) within frames transmitted during the CFP
1–16 383	0	1	Reserved
0	1	1	Reserved
1–2007	1	1	AID in PS-Poll frames
2008–16 383	1	1	Reserved

Table 2.4 - Duration/ID field

To control medium access, wireless devices need to monitor packet headers and update the NAV (Network Allocation Network). Therefore, when bit 15 in the Duration/ID field equals 0, the value in this field indicates the time (in microseconds) in which the NAV must be updated. For packets transmitted during Contention Free Periods (CFP) the default actualization time is 32,768 microseconds.

For PS-Poll packets, the Duration/ID field has another interpretation, since this packet type is used by mobile devices to retrieve their AP stored packets; AID (Association ID) indicating the BSS to which the device belongs is obtained from the Duration/ID field.

2.3.3 Address

There can be up to 4 Address fields in a MAC Header; each one may serve a different purpose depending on the frame type. Addresses are composed of 48 bits, according to IEEE 802 standard and were designed to identify a device, group of devices or all devices that make up a network. Addresses can be utilized for distinct finalities, as detailed here:

- DA (Destination Address). An IEEE MAC identifier of 48 bits that denotes the MAC entity or entities that denotes the final destination, which will manage the MSDU for protocol processing on the higher layers.
- SA (Source Address). An IEEE MAC of 48 bits identifies the MAC entity from which the transfer of the MSDU contained in the

frame body field was initiated.

- RA (Receiver Address) Single our group 48 bit address that indicates the intended immediate recipient STAs, on the WM, for the information contained in the frame body field.
- TA (Transmitter Address). A 48 bits IEEE MAC address identifying the station that has transmitted de 48 bits, onto the WM, the MPDU contained in the frame body field. In order words, the wireless device that transmitted the packet over the wireless medium.
- BSSID (*Basic Service Set ID*). MAC address that enunciates the Wireless LAN to which the device has been assigned. In ad-hoc networks, a random BSSID is generated –according to regulations on IEEE 802- to avoid conflicts with legal MAC addresses. For networks with infrastructure, the BSSID is the MAC address of the Access Point.

Since Address fields use depends on frame type and subtype, most frames utilize three fields for DA, SA and BSSID. However in Data packets, field use depends on the existing network.

2.3.4 Sequence Control

Sequence Control is a 16 bits field, used in the process of defragmentation that helps eliminate packet duplication, utilizing Fragment Number and Sequence Number subfields (illustrated in Figure 2.9).

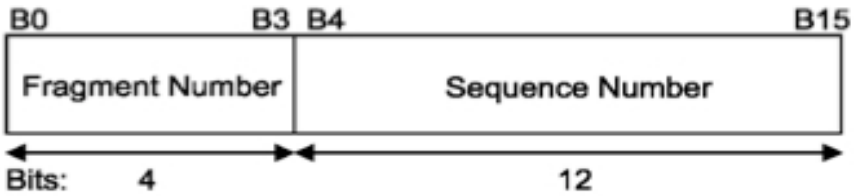


Figure 2.10 - Sequence Control field

The subfield Fragment Number allows to control packet re-assembly, by creating a numerical identifier (4 bits) that acquires value 0 for the first fragment and increases in a +1 for each successive fragment. This identifier remains constant in every re-transmission of fragments.

In the management of higher level packets on the MAC sublayer, each packet is assigned a sequential number. This value is stored in the Sequence Number subfield (12 bits). Value assignation happens through a module 4096 counter, where the first packet is assigned Sequence Number value 0 and subsequent packets increase +1 in value. This value remains unaltered in re-transmissions, as do the fragments.

2.3.5 QoS (Quality of Service) Control Field

This is 16 bits field that identifies the TC or TS to which the frame belongs to, it also displays information relative to the QoS. The field is present in data frames with a fixed value of 1 (as detailed on section 3.2.1). Each QoS Control Field is composed by 5 sub-parts that depend on the designated sender (HC or no –AP STA) and on the frame type and subtype. Table 2.5 displays this information.

Applicable frame (sub) types	Bits 0–3	Bit 4	Bits 5-6	Bit 7	Bits 8–15
QoS (+)CF-Poll frames sent by HC	TID	EOSP	Ack Policy	Reserved	TXOP Limit
QoS Data, QoS Null, and QoS Data+CF-Ack frames sent by HC	TID	EOSP	Ack Policy	Reserved	AP PS Buffer State
QoS data frames sent by non-AP STAs	TID	0	Ack Policy	Reserved	TXOP Duration Requested
	TID	1	Ack Policy	Reserved	Queue Size

Table 2.5 - QoS Control field

2.3.6 Frame Body

A variable length frame, it contents specific information to individual frame types and subtypes. Maximum capacity is determined by maximum length (MSDU+ICV+IV), where ICV (Integrity Check Value) y el IV (Initialization Vector) are subfields corresponding to the WEP service (Section 2.3.1).

2.3.7 FCS

This field contains 32 bits CRC (Cyclic Redundancy Code) that verifies frame integrity. The MAC Header and the Frame Body are used to calculate the CRC; these two fields are commonly referred to as calculation fields.

The FCS is calculated using the following standard generator polynomial of degree 32:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1 \qquad (1.1)$$

The FCS is the ones complement of the sum (module 2) of the following:

a) The remainder of $x^{k} \times (x^{31} + x^{30} + x^{29} + \ldots + x^{2} + x + 1)$ divided (module2) by $G(x)$, where k is the number of bits in the calculation fields, and

b) The remainder after multiplication of the contents (treated as a polynomial) of the calculation fields by $x32$ and then division by $G(x)$.

The FCS field is transmitted commencing with the coefficient of the highest-order term.

According to the IEEE 802.11 Standard, with the FCS value of a transmitted packet, receiving devices can verify if the packet was altered during transmission by comparing it to the calculated CRC. If the CRC correspond to the packet, an affirmative acknowledgement packet in sent to the transmitting device. If the CRC does not match the packet, waiting time will end and the transmitter will need to retransmit. In 802.11 there are no negative acknowledgements.

2.3.8 Format for individuals frame types

In prior sections fields that define frame format have been explained; however the presence or absence of these fields and their specifications depend on frame type and subtype.

2.3.8.1 Control frames

The subfields composing Frame Control correspond to those displayed on Figure 2.8; the remainder of the structure depends on the specific sub-frame type being used. Here are some examples of frame subtypes with their structures. (Please refer to the Standard for more detailed information).

2.3.8.1.1 RTS (Request to Send)

Request to Send is represent by a MAC Header, containing the subfields Frame Control, Duration, RA and TA, as well as the FCS field. There is no frame body in this case.

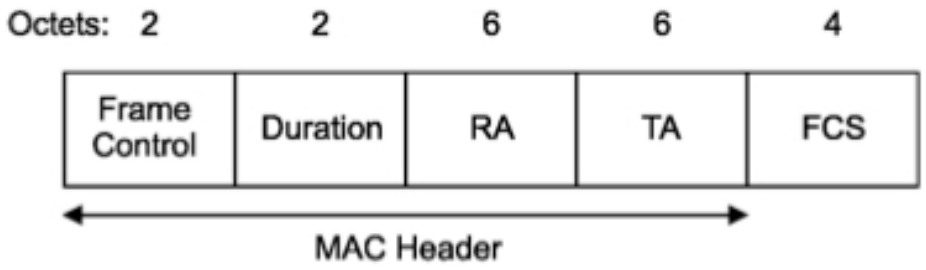


Figure 2.11 - RTS frame structure

Duration field will depend on the presence or absence of QoS capability on the station as well as method for medium access. The RA field of the RTS frame is the address of the STA, on the WM, that is the intended immediate recipient of the pending directed data or management frame.

The TA field is the address of the STA transmitting the RTS frame.

2.3.8.1.2 CTS (Clear to Send)

Clear to Send structure is also free of Frame Body.

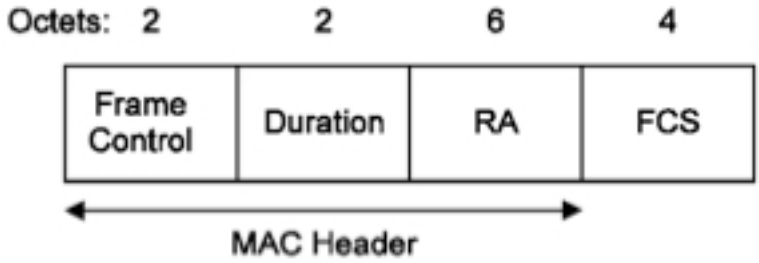


Figure 2.12 - CTS frame structure.

When the CTS frame follows an RTS frame, the RA field of the CTS frame is copied from the TA field of the immediately previous RTS frame to which the CTS is a response. When the CTS is the first frame in a frame exchange, the RA field is set to the MAC address of the transmitter.

For all CTS frames sent in response to RTS frames, the duration value is the value obtained from the Duration field of the immediately previous RTS frame, minus the time, in

microseconds, required to transmit the CTS frame and its SIFS interval. If the calculated duration includes a fractional microsecond, that value is rounded up to the next higher integer.

Specifications for the Duration field are detailed in extent on the Standard.

2.3.8.1.3 ACK (Acknowledgment)

A very similar structure to its predecessors, the difference is that RA field is copied from the Address 2 field of the immediately previous directed data, management, BlockAckReq control, BlockAck control, or PS-Poll control frame.

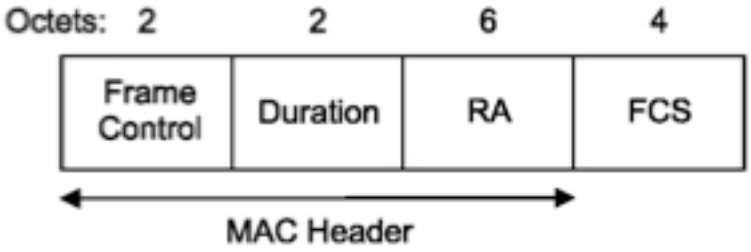


Figure 2.13 - ACK frame structure

2.3.8.1.4 PS-Poll

In this case, in correspondence with the Duration field, we find the AID (Association Identifier), defined as the value assigned to the STA transmitting the frame by the AP in the association response frame that established that STA's current association.

The BSSID is the address of the STA contained in the AP. The TA field is the address of the STA transmitting the frame.

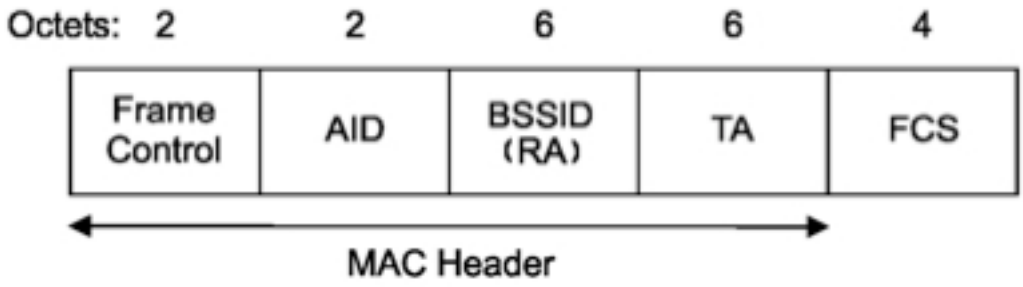


Figure 2.14 - PS-Poll frame structure

2.3.8.1.5 CF-End

For the Contention Free-End format, Duration field is fixed to 0, while the RA field is the broadcast group address and the BSSID field is the address of the STA contained in the AP.

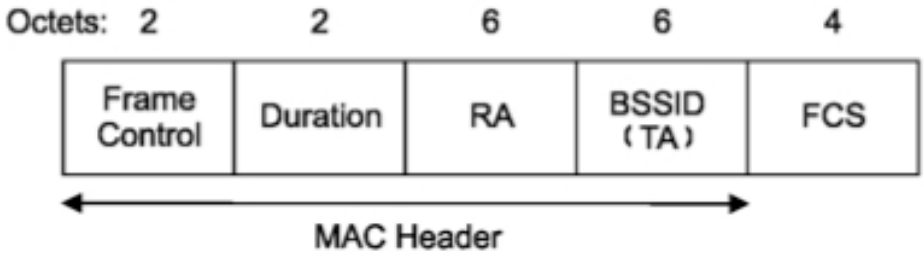


Figure 2.15 - CF-End frame structure

2.3.8.2 Data Frames

Data frames are illustrated in figure 2.7. The fields Address 1, 2, 3 and 4 depend on the values To DS and From DS. These relationships are reported on Table 2.5:

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	RA = DA	TA = SA	BSSID	N/A
0	1	RA = DA	TA = BSSID	SA	N/A
1	0	RA = BSSID	TA = SA	DA	N/A
1	1	RA	TA	DA	SA

Table 2.6 - Valid values for addresses field

The frame body consists of the MSDU, or a fragment thereof, and a security header and trailer (if and only if the Protected Frame subfield in the Frame Control field is set to 1). The frame body is null (0 octets in length) in data frames of subtype Null (no data), CF-ACK (no data), CF-Poll (no data), and CF-Ack+CF-Poll (no data), regardless of the encoding of the QoS subfield in the Frame Control field.

For data frames of subtype Null (no data), CF-ACK (no data), CF-Poll (no data), and CF-Ack+CF-Poll (no data) and for the corresponding QoS data frame subtypes, the

Frame Body field is omitted; these subtypes are used for MAC control purposes. For data frames of subtypes Data, Data+CF-Ack, Data+CF-Poll, and Data+CF-Ack+CF+Poll and for the corresponding four QoS data frame subtypes, the Frame Body field contains all of, or a fragment of, an MSDU after any encapsulation for security.

Calculations for the duration field depend on factors such as utilized access method, specifications for the calculation are provided in the standard.

2.3.8.3 Management Frames

Management Frames are structured as follows, with independence of subtypes:

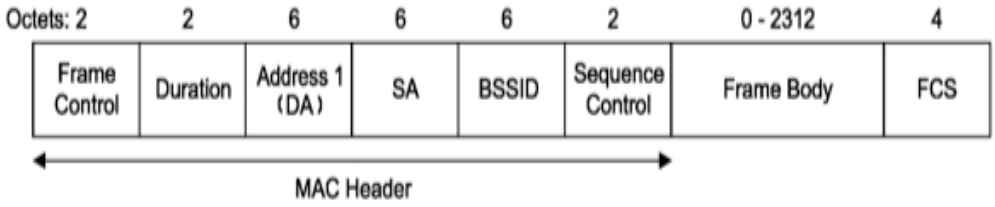


Figure 2.16 - Management frame structure

Figure 2.16 displays how Address 1 corresponds to the Destination Address (in this case, the STAuses the contents of this field to perform the address matching for receive decisions). Address 2 refers to the Source Address (SA) indicating the address of the STA transmitting the frame.

On the other hand, the BSSID of the management frame is determined as follows:

a) If the STA is an AP or is associated with an AP, the BSSID is the address currently in use by the STA contained in the AP.

b) If the STA is a member of an IBSS, the BSSID is the BSSID of the IBSS.

The address fields for management frames do not vary by frame subtype.

Regarding the Duration field, values will depend on the method of access and the presence or absence of QoS capability.

Frame body field will be specific to the frame subtype employed. The table displays all Control Frame subtypes. We will explain in detail some Control Frames of interest to this project.

2.3.8.3.1 Beacon Frame

This frame type is relevant to us because it contains all the information about the network. Beacon frames are transmitted periodically to announce the presence of a Wireless LAN network. Beacon frames are transmitted by the Access Point (AP) in an infrastructure BSS. In IBSS network beacon generation is distributed among the stations.

Tables 2.7. and 2.78 display all the components of Frame Body, such as Beacon Interval, Timestamp (both vital for timing and synchronization) and others like Supported rates, Frequency Hopping Parameters, Direct Sequence Parameter (that reveal characteristics of the PHY layer).

Order	Information	Notes
1	Timestamp	
2	Beacon interval	
3	Capability	
4	Service Set Identifier (SSID)	
5	Supported rates	
6	Frequency-Hopping (FH) Parameter Set	The FH Parameter Set information element is present within Beacon frames generated by STAs using FH PHYs.
7	DS Parameter Set	The DS Parameter Set information element is present within Beacon frames generated by STAs using Clause 15, Clause 18, and Clause 19 PHYs.
8	CF Parameter Set	The CF Parameter Set information element is present only within Beacon frames generated by APs supporting a PCF.
9	IBSS Parameter Set	The IBSS Parameter Set information element is present only within Beacon frames generated by STAs in an IBSS.

Table 2.7 - Beacon frame body (1)

Order	Information	Notes
10	Traffic indication map (TIM)	The TIM information element is present only within Beacon frames generated by APs.
11	Country	The Country information element shall be present when dot11MultiDomainCapabilityEnabled is true or dot11SpectrumManagementRequired is true.
12	FH Parameters	FH Parameters as specified in 7.3.2.10 may be included if dot11MultiDomainCapabilityEnabled is true.
13	FH Pattern Table	FH Pattern Table information as specified in 7.3.2.11 may be included if dot11MultiDomainCapabilityEnabled is true.
14	Power Constraint	Power Constraint element shall be present if dot11SpectrumManagementRequired is true.
15	Channel Switch Announcement	Channel Switch Announcement element may be present if dot11SpectrumManagementRequired is true.
16	Quiet	Quiet element may be present if dot11SpectrumManagementRequired is true.
17	IBSS DFS	IBSS DFS element shall be present if dot11SpectrumManagementRequired is true in an IBSS.
18	TPC Report	TPC Report element shall be present if dot11SpectrumManagementRequired is true.
19	ERP Information	The ERP Information element is present within Beacon frames generated by STAs using extended rate PHYs (ERPs) defined in Clause 19 and is optionally present in other cases.
20	Extended Supported Rates	The Extended Supported Rates element is present whenever there are more than eight supported rates, and it is optional otherwise.
21	RSN	The RSN information element shall be present within Beacon frames generated by STAs that have dot11RSNAEnabled set to TRUE.
22	BSS Load	The BSS Load element is present when dot11QosOption-Implemented and dot11QBSSLoadImplemented are both true.
23	EDCA Parameter Set	The EDCA Parameter Set element is present when dot11QosOptionImplemented is true and the QoS Capability element is not present.
24	QoS Capability	The QoS Capability element is present when dot11QosOption-Implemented is true and EDCA Parameter Set element is not present.
Last	Vendor Specific	One or more vendor-specific information elements may appear in this frame. This information element follows all other information elements.

Table 2.8 - Beacon frame body (2)

2.3.8.3.2 Association Request

Through this type of message the STA requires association to an AP. This message contains in its frame body a series of compartments used to indicate requested or advertised capabilities.

Order	Information	Notes
1	Capability	
2	Listen interval	
3	SSID	
4	Supported rates	
5	Extended Supported Rates	The Extended Supported Rates element is present whenever there are more than eight supported rates, and it is optional otherwise.
6	Power Capability	The Power Capability element shall be present if dot11SpectrumManagementRequired is true.
7	Supported Channels	The Supported Channels element shall be present if dot11SpectrumManagementRequired is true.
8	RSN	The RSN information element is only present within Association Request frames generated by STAs that have dot11RSNAEnabled set to TRUE.
9	QoS Capability	The QoS Capability element is present when dot11QosOption-Implemented is true.
Last	Vendor Specific	One or more vendor-specific information elements may appear in this frame. This information element follows all other information elements.

Table 2.9 - Association Request frame body

2.3.8.3.3 Association Response

Once the Request is placed, the AP replies with an Association Response message stating whether the request was accepted. If it was, it sends as a message field the AID (Association Identifier), a 16 bits compartment that represents the STA identification. The following tables illustrate the full Frame Body of the Association Response frame.

Order	Information	Notes
1	Capability	
2	Status code	
3	AID	

Table 2.10 - Association Response frame body

(1)

Order	Information	Notes
4	Supported rates	
5	Extended Supported Rates	The Extended Supported Rates element is present whenever there are more than eight supported rates, and it is optional otherwise.
6	EDCA Parameter Set	
Last	Vendor Specific	One or more vendor-specific information elements may appear in this frame. This information element follows all other information elements.

Table 2.11 - Association Response frame body (2)

2.3.8.3.4 Probe Request

A station sends a probe request frame when it needs to obtain information from another station.

Order	Information	Notes
1	SSID	
2	Supported rates	
3	Request information	May be included if dot11MultiDomainCapabilityEnabled is true.
4	Extended Supported Rates	The Extended Supported Rates element is present whenever there are more than eight supported rates, and it is optional otherwise.
Last	Vendor Specific	One or more vendor-specific information elements may appear in this frame. This information element follows all other information elements.

Table 2.12 - Probe Request frame body

2.3.8.3.5 Probe Response

A station will respond with a probe response frame, containing capability information, supported data rates, etc., when after it receives a probe request frame.

Order	Information	Notes
1	Timestamp	
2	Beacon interval	
3	Capability	
4	SSID	
5	Supported rates	
6	FH Parameter Set	The FH Parameter Set information element is present within Probe Response frames generated by STAs using FH PHYs.
7	DS Parameter Set	The DS Parameter Set information element is present within Probe Response frames generated by STAs using Clause 15, Clause 18, and Clause 19 PHYs.
8	CF Parameter Set	The CF Parameter Set information element is present only within Probe Response frames generated by APs supporting a PCF.
9	IBSS Parameter Set	The IBSS Parameter Set information element is present only within Probe Response frames generated by STAs in an IBSS.
10	Country	Included if dot11MultiDomainCapabilityEnabled or dot11SpectrumManagementRequired is true.
11	FH Parameters	FH Parameters, as specified in 7.3.2.10, may be included if dot11MultiDomainCapabilityEnabled is true.
12	FH Pattern Table	FH Pattern Table information, as specified in 7.3.2.11, may be included if dot11MultiDomainCapabilityEnabled is true.
13	Power Constraint	Shall be included if dot11SpectrumManagementRequired is true.
14	Channel Switch Announcement	May be included if dot11SpectrumManagementRequired is true.
15	Quiet	May be included if dot11SpectrumManagementRequired is true.
16	IBSS DFS	Shall be included if dot11SpectrumManagementRequired is true in an IBSS.
17	TPC Report	Shall be included if dot11SpectrumManagementRequired is true.
18	ERP Information	The ERP Information element is present within Probe Response frames generated by STAs using ERPs and is optionally present in other cases.
19	Extended Supported Rates	The Extended Supported Rates element is present whenever there are more than eight supported rates, and it is optional otherwise.
20	RSN	The RSN information element is only present within Probe Response frames generated by STAs that have dot11RSNA-Enabled set to TRUE.
21	BSS Load	The BSS Load element is present when dot11QosOption-Implemented and dot11QBSSLoadImplemented are both true.
22	EDCA Parameter Set	The EDCA Parameter Set element is present when dot11QosOptionImplemented is true.
Last–*l*	Vendor Specific	One or more vendor-specific information elements may appear in this frame. This information element follows all other information elements, except the Requested Information elements.
Last–*n*	Requested information elements	Elements requested by the Request information element of the Probe Request frame.

Table 2.13 - Probe Response frame body

2.4 MAC Sublayer Functional Description

Figure 2.17 shows the architecture of the MAC sublayer, including the distributed coordination function (DCF), the point coordination function (PCF), the hybrid coordination function (HCF) and their coexistence in an IEEE 802.11 LAN.

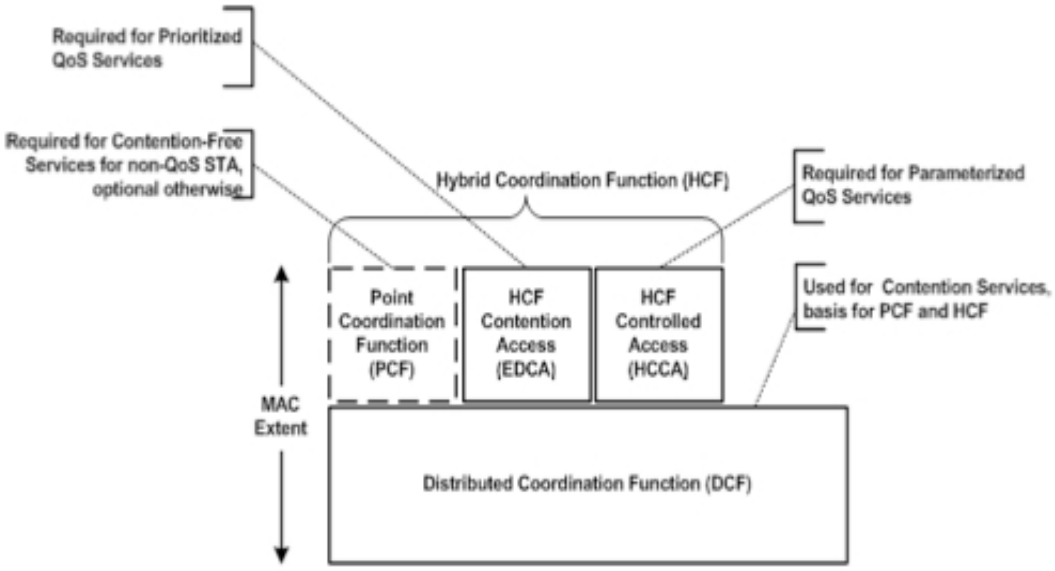


Figure 2.17 - MAC architecture

2.4.1 MAC Architecture

The basic 802.11 MAC layer uses the Distributed Coordination Function (DCF) to share the medium between multiple stations. DCF relies on CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) and optional 802.11 RTS/CTS to share the medium between stations. The STA that wants to transmit needs to sense the medium to determine is another station is transmitting, if the medium is available transmission can be initiated. If the medium is busy, transmission is delayed until the ongoing is finished. After wait time, the STA must select a random backoff interval and shall decrement the backoff interval counter while the medium is idle. Refinement of the method can minimize future

collisions, through an interchange of exchange short control frames (Request to Send-Clear to Send).

The original 802.11 MAC defines another coordination function called the Point Coordination Function (PCF): this is available only in "infrastructure" mode, where stations are connected to the network through an Access Point (AP). This is a polling operation, where Point Coordination (PC), operating as AP of the BSS exerts control as Poll master. PCF distributes information internally to the Beacon frames to control the medium and fix the Network Allocation Vectors (NAV) of the stations. Additionally, all frames transmitted during PCF can use smaller Inter Frame Spaces (IFS) than they will through the DFC, gaining priority in medium access.

Access priority provided by a PCF reflects the creation of a Contention Free (CF) access method. Where the PC both controls frame transmission and eliminates contention for a limited period of time.

DCF and PCF shall coexist in way in which can operate contemporaneously on the same BSS.

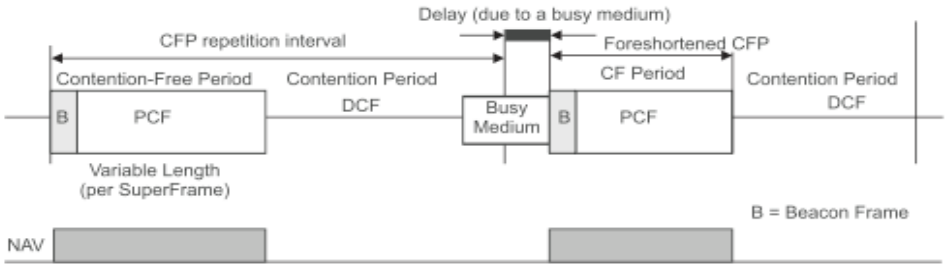


Figure 2.18 - Coexistence Contention Free and Contention Period

When an HC is operating in a BSS, it may generate an alternation of CFP and CP in the same way as a PC, using the DCF access method only during the CP. The HCF access methods (controlled and contention-based) operate sequentially when the channel is in CP.

The IEEE 802.11, 2007 Edition enhances the DCF and the PCF, through a new coordination function: the Hybrid Coordination Function (HCF). This additional coordination function is supported only on QoS network configuration must be implemented in each QoS station. The HCF combines DCF and PCF functions with some improvements, QoS specific mechanism and a series of sub frames that allow uniform

interchange of sequences for transmission of QoS data during either CF or CFP.

Within the HCF, there are two methods of channel access, similar to those defined in the legacy 802.11 MAC: HCF Controlled Channel Access (HCCA) and Enhanced Distributed Channel Access (EDCA). Both EDCA and HCCA define Traffic Categories (TC). For example, emails could be assigned to a low priority class, and Voice over Wireless LAN (VoWLAN) could be assigned to a high priority class.

With EDCA (Enhanced Distributed Channel Access), high priority traffic has a higher chance of being sent than low priority traffic: a station with high priority traffic waits a little less before it sends its packet, on average, than a station with low priority traffic. In addition, each priority level is assigned a Transmit Opportunity (TXOP). A TXOP is a bounded time interval during which a station can send as many frames as possible (as long as the duration of the transmissions does not extend beyond the maximum duration of the TXOP). If a frame is too large to be transmitted in a single TXOP, it should be fragmented into smaller frames. The use of TXOPs reduces the problem of low rate stations gaining an inordinate amount of channel time in the legacy 802.11 DCF MAC. A TXOP time interval of 0 means it is limited to a single MSDU or MMPDU.

According to the IEEE 802.11 Standard, the purpose of QoS is to protect high priority data from low priority data but there can be scenarios in which the data which belongs to same priority needs to be protected from data of same priority. Example being suppose a network can accommodate only 10 data calls & an eleventh call is made. Admission Control in EDCA addresses this type of problems. The AP publishes the available bandwidth in beacons. The clients can check the available bandwidth before adding more traffic in the network that cannot be entertained.

Wi-Fi Multimedia (WMM) certified APs must be enabled for EDCA and TXOP. All other enhancements of the 802.11e amendment are optional.

Instead, the HCCA (HCF (Hybrid Coordination Function) Controlled Channel Access) works a lot like the Point Coordination Function. However, in contrast to PCF, in which the interval between two beacon frames is divided into two periods of CFP and CP, the HCCA allows for CFPs being initiated at almost any time during a CP. This kind of CFP is called a Controlled Access Phase (CAP) in 802.11e. A CAP is initiated by the AP, whenever it wants to send a frame to a station, or receive a frame from a station, in a

contention free manner. In fact, the CFP is a CAP too. During a CAP, the Hybrid Coordinator (HC) – which is also the AP – controls the access to the medium. During the CP, all stations function in EDCA. The other difference with the PCF is that Traffic Class (TC) and Traffic Streams (TS) are defined. This means that the HC is not limited to per-station queuing and can provide a kind of per-session service. Also, the HC can coordinate these streams or sessions in any fashion it chooses (not just round-robin). Moreover, the stations give info about the lengths of their queues for each Traffic Class (TC). The HC can use this info to give priority to one station over another, or better adjust its scheduling mechanism. Another difference is that stations are given a TXOP: they may send multiple packets in a row, for a given time period selected by the HC. During the CP, the HC allows stations to send data by sending CF-Poll frames.

HCCA is generally considered the most advanced (and complex) coordination function. With the HCCA, QoS can be configured with great precision. QoS-enabled stations have the ability to request specific transmission parameters (data rate, jitter, etc.) which should allow advanced applications like VoIP and video streaming to work more effectively on a Wi-Fi network. HCCA support is not mandatory. In fact, few (if any) APs currently available are enabled for HCCA. Nevertheless, implementing the HCCA does not require much overhead, as it basically uses the existing DCF mechanism for channel access (no change to DCF or EDCA operation is needed). In particular, the station side implementation is very simple as stations only need to be able to respond to poll messages. On the AP side, however, a scheduler and queuing mechanism is needed. Given that AP's are already equipped better than station transceivers, this should not be a problem either.

2.4.2 DCF

As previously explained, DCF is the fundamental MAC technique of the IEEE 802.11 based WLAN standard. DCF employs a CSMA/CA with Binary exponential backoff algorithm. Also, all directed traffic uses immediate positive acknowledgment (ACK frame) in those cases when retransmission is scheduled by the sender if no ACK is received.

The Carrier Sense Multiple Access (CSMA) is a probabilistic Media Access Control (MAC) protocol in which a STA verifies the absence of other traffic before transmitting on a shared transmission medium, such as an electrical bus, or a band of the electromagnetic spectrum.

"Carrier Sense" describes the fact that a transmitter listens for a carrier wave before trying to send. That is, it tries to detect the presence of an encoded signal from another station before attempting to transmit. If a carrier is sensed, the station waits for the transmission in progress to finish before initiating its own transmission.

The CA functionality CA (Collision Avoidance) is modification of pure Carrier Sense Multiple Access and decreases probability of collisions wherever those are most likely to occur, specifically when the medium goes from busy to ideal state. This is the kind of situation that needs a random backoff procedure to troubleshoot.

Carrier sense shall be performed through physical and virtual mechanisms. Virtual is achieved through distribution of reservation information that indicates medium use. The Request to Send (RTS) - Clear to Send (CTS) interchange, prior to data frame transmission is the most common method to distribute this information to the medium. The RTS and CTS frames contain a Duration field that defines the period of time that the medium is to be reserved to transmit the actual data frame and the returning ACK frame. All STAs within the reception range of either the originating STA (which transmits the RTS) or the destination STA (which transmits the CTS) shall learn of the medium reservation. Thus, a STA can be unable to receive from the originating STA and yet still know about the impending use of the medium to transmit a data frame.

The RTS/CTS mechanism cannot be used for MPDUs with broadcast and multicast immediate destination because there are multiple recipients for the RTS, and thus potentially multiple concurrent senders of the CTS in response. The RTS/CTS mechanism need not be used for every data frame transmission. Because the additional RTS and CTS frames add overhead inefficiency, the mechanism is not always justified, especially for short data frames.

A STA configured not to initiate the RTS/CTS mechanism shall still update its virtual CS mechanism with the duration information contained in a received RTS or CTS frame, and shall always respond to an RTS addressed to it with CTS if permitted by medium access rules.

To support the proper operation of the RTS/CTS and the virtual CS mechanism, all STAs shall be able to detect the RTS and CTS frames.

Virtual carrier sensing must be under control of the MAC sublayer, and related

closely to the network allocation vector (NAV). The NAV maintains a prediction of future traffic on the medium based on duration information that is announced in RTS/CTS frames prior to the actual exchange of data.

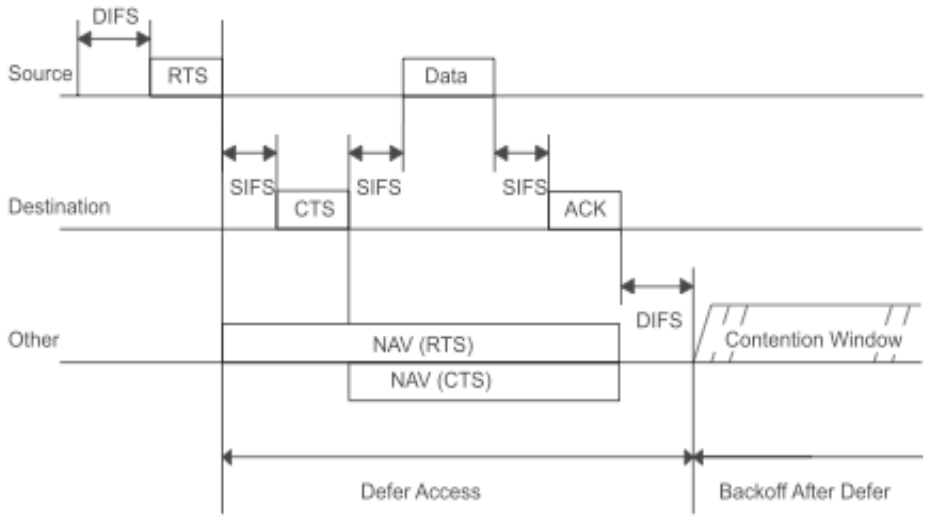


Figure 2.19 - RTS-CTS procedure

STAs receiving a valid frame shall update their NAV with the information received in the Duration field for all frames where the new NAV value is greater than the current NAV value, except the NAV shall not be updated where the RA is equal to the receiving STA's MAC address. Upon receipt of a PS-Poll frame, a STA shall update its NAV settings as appropriate under the data rate selection rules using a duration value equal to the time, in microseconds, required to transmit one ACK frame plus one SIFS interval, but only when the new NAV value is greater than the current NAV value. If the calculated duration includes a fractional microsecond, that value is rounded up the next higher integer.

Figure 2.19 shows the NAVs in two stations, the longest bar corresponds to the station that received the RTS frame and the shorter to the recipient of the CTS, length of these vectors depends on the values on the Duration field.

2.4.2.1 Inter Frame Space (IFS)

To provide access priority levels to the medium, five types of Inter Frame Spaces are defined:

- ✓ SIFS (Short InterFrame Space)

As shown on figure 2.20 , this is the shortest of the IFSs. It's used to provide

an efficient MSDU delivery mechanism. The SIFS shall be used prior to transmission of an ACK frame, a CTS frame, the second or subsequent MPDU of a fragment burst, and by a STA responding to any polling by the PCF. The SIFS may also be used by a PC for any types of frames during the CFP .The SIFS is the time from the end of the last symbol of the previous frame to the beginning of the first symbol of the preamble of the subsequent frame as seen at the air interface. SIFS shall be used when STAs have seized the medium and need to keep it for the duration of the frame exchange sequence to be performed. Using the smallest gap between transmissions within the frame exchange sequence prevents other STAs, which are required to wait for the medium to be idle for a longer gap, from attempting to use the medium, thus giving priority to completion of the frame exchange sequence in progress.

Once the STA has contended for the channel, that STA shall continue to send fragments until either all fragments of a single MSDU or MMPDU have been sent, an acknowledgment is not received, or the STA is restricted from sending any additional fragments due to a dwell time boundary.

✓ PIFS (Point InterFrame Space)

The PIFS shall be used only by STAs operating under the PCF to gain priority access to the medium at the start of the CFP or by a STA to transmit a Channel Switch Announcement frame. A STA using the PCF shall be allowed to transmit CF traffic after its CS mechanism determines that the medium is idle at the TxPIFS slot boundary.

✓ DIFS (DCF Interframe Space)

The DIFS shall be used by STAs operating under the DCF to transmit data frames (MPDUs) and management frames (MMPDUs). A STA using the DCF shall be allowed to transmit if its CS mechanism determines that the medium is idle at the TxDIFS slot boundary after a correctly received frame, and its backoff time has expired.

✓ AIFS (Arbitration Interframe Space)

The AIFS shall be used by QoS STAs to transmit all data frames (MPDUs), all management frames (MMPDUs), and the following control frames: PS-Poll, RTS, CTS (when not transmitted as a response to the RTS), BlockAckReq, and BlockAck (when not transmitted as a response to the

BlockAckReq).

✓ EIFS (Extended Interframe Space)

A STA's DCF shall use EIFS before transmission, when it determines that the medium is idle following reception of a frame for which the PHY-RXEND.indication primitive contained an error or a frame for which the MAC FCS value was not correct. The EIFS is defined to provide enough time for another STA to acknowledge what was, to this STA, an incorrectly received frame before this STA commences transmission.

The different IFSs shall be independent of the STA bit rate. The IFS timings are defined as time gaps on the medium, and the IFS timings except AIFS are fixed for each PHY (even in multirate-capable PHYs). The IFS values are determined from attributes specified by the PHY.

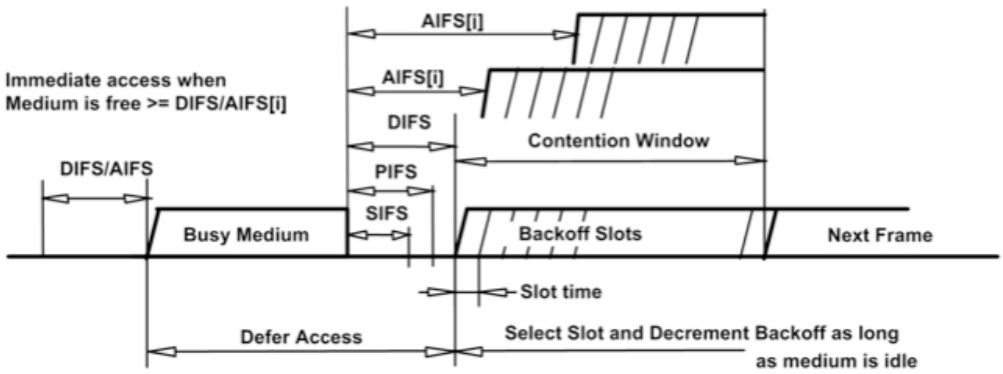


Figure 2.20 - IFS relationships

2.4.2.2 Backoff procedure

Whenever an STA needs to initiate a MPDU or MMPUD transmission it needs to activate the carrier sense mechanism to determine the busy/idle state of the medium. If the medium is busy, the STA shall defer until the medium is determined to be idle without interruption for a period of time equal to DIFS when the last frame detected on the medium was received correctly, or after the medium is determined to be idle without interruption for a period of time equal to EIFS when the last frame detected on the medium was not received correctly. After this DIFS or EIFS medium idle time, the STA shall then generate a random backoff period for an additional deferral time before transmitting, unless the backoff timer already contains a nonzero value, in which case the selection of a random number is not

needed and not performed. This process minimizes collisions during contention between multiple STAs that have been deferring to the same event. Waiting time is determined through this equation:

$$\text{Backoff Time} = \text{Random}() \times \text{aSlotTime} \qquad (1.2)$$

where:

Random() = Pseudo-random integer drawn from a uniform distribution over the interval [0,CW].

aSlotTime = The value of the correspondingly named PHY characteristic.

The contention window (CW) parameter shall take an initial value of aCWmin. Every STA shall maintain a STA short retry count (SSRC) as well as a STA long retry count (SLRC), both of which shall take an initial value of zero. The SSRC shall be incremented when any short retry count (SRC) associated with any MPDU of type Data is incremented. The SLRC shall be incremented when any long retry count (LRC) associated with any MPDU of type Data is incremented. The CW shall take the next value in the series every time an unsuccessful attempt to transmit an MPDU causes either STA retry counter to increment, until the CW reaches the value of aCWmax. A retry is defined as the entire sequence of frames sent, separated by SIFS intervals, in an attempt to deliver an MPDU. Once it reaches aCWmax, the CW shall remain at the value of aCWmax until the CW is reset. This improves the stability of the access protocol under high-load conditions.

2.4.2.3 Basic Access

The following is an illustration of the process through which an STA determines if it is able to transmit through the wireless medium.

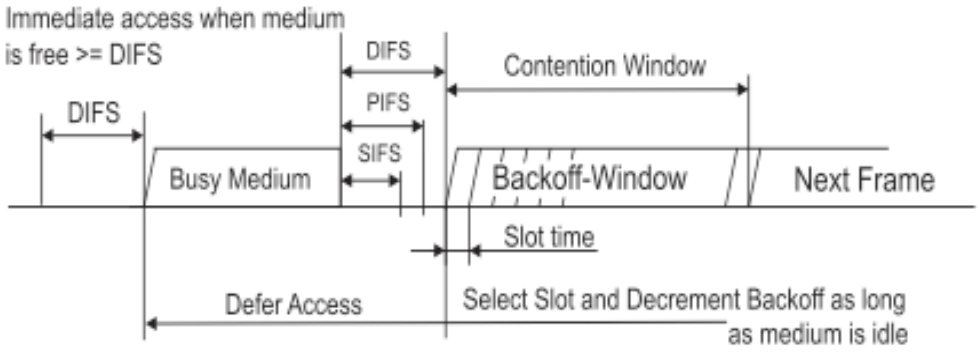


Figure 2.21 - Basic Access method

As displayed on figure 2.21, a STA may transmit a pending MPDU when it is operating under the DCF access method, either in the absence of a PC, or in the CP of the PCF access method, when the STA determines that the medium is idle for greater than or equal to a DIFS period, or an EIFS period if the immediately preceding medium-busy event was caused by detection of a frame that was not received at this STA with a correct MAC FCS value. If, under these conditions, the medium is determined by the CS mechanism to be busy when a STA desires to initiate the initial frame of one of the frame exchanges, exclusive of the CF period, the random backoff procedure shall be followed.

To kick start the backoff procedure, the STA shall set its Backoff Timer to a random backoff time using the equation shown on section 2.3.2.2. How displays the figure 2.22, all backoff slots occur following a DIFS period during which the medium is determined to be idle for the duration of the DIFS period, or following an EIFS period during which the medium is determined to be idle for the duration of the EIFS period, as appropriate.

A STA performing the backoff procedure shall use the CS mechanism to determine whether there is activity during each backoff slot. If no medium activity is indicated for the duration of a particular backoff slot, then the backoff procedure shall decrement its backoff time by aSlotTime.

If the medium is determined to be busy at any time during a backoff slot, then the backoff procedure is suspended; that is, the backoff timer shall not decrement for that slot. Transmission shall commence when the Backoff Timer reaches zero.

In the case of unsuccessful transmissions requiring acknowledgment, this backoff procedure shall begin at the end of the ACKTimeout interval.

The effect of this procedure is that when multiple STAs are deferring and go into random backoff, then the STA selecting the smallest backoff time using the random function will win the contention (assuming all of the contending STAs detect the same instances of WM activity at their respective receivers).

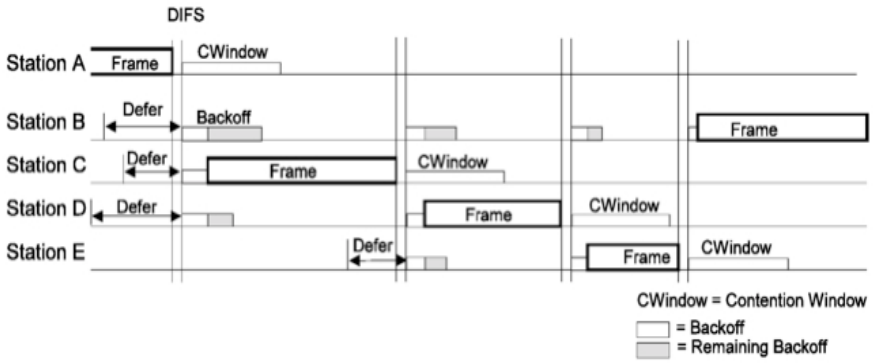


Figure 2.22 - Backoff procedure

A STA that is addressed by an RTS frame shall transmit a CTS frame after a SIFS period if the NAV at the STA receiving the RTS frame indicates that the medium is idle. If the NAV at the STA receiving the RTS indicates the medium is not idle, that STA shall not respond to the RTS frame. The Duration field in the CTS frame shall be the duration field from the received RTS frame, adjusted by subtraction of aSIFSTime and the number of microseconds required to transmit the CTS frame at a data rate determined.

Upon successful reception of a frame of a type that requires acknowledgment with the To DS field set, an AP shall generate an ACK frame. After a successful reception of a frame requiring acknowledgment, transmission of the ACK frame shall commence after a SIFS period, without regard to the busy/idle state of the medium.

The basic access mechanism is illustrated at figure 2.23:

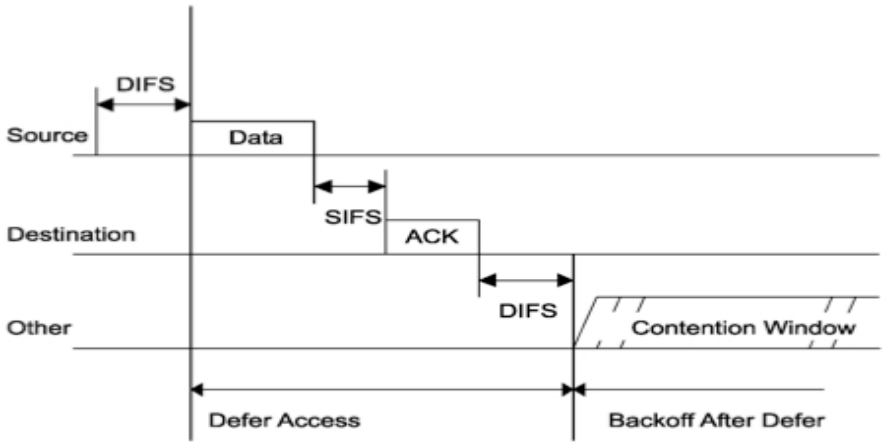


Figure 2.23 - ACK procedure

2.4.3 PCF

Each CFP shall begin with a Beacon frame that contains a DTIM element (Table2.8). The CFPs shall occur at a defined repetition rate, which shall be synchronized with the beacon interval.

The PC generates CFPs at the CFP repetition interval (CFPPeriod), which is defined as a number of DTIM intervals. The PC shall determine the CFPPeriod (depicted as a repetition interval in the illustrations in Figure 2.24) to use from the CFPPeriod parameter in the CF Parameter Set. This value, in units of DTIM intervals, shall be communicated to other STAs in the BSS in the CFPPeriod field of the CF Parameter Set element of Beacon frames. The CF Parameter Set element shall only be present in Beacon and Probe Response frames transmitted by STAs containing an active PC.

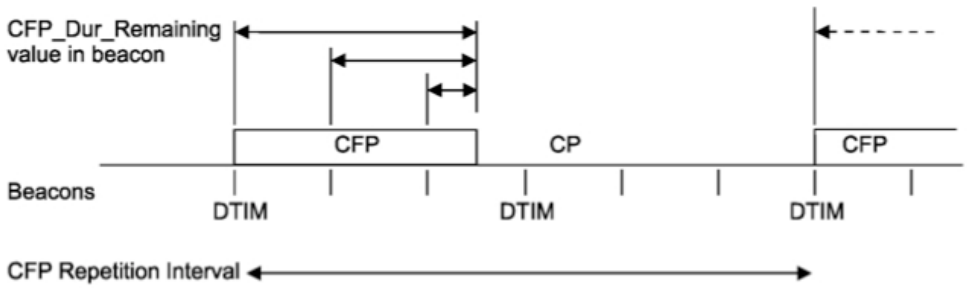


Figure 2.24 - Timing relationships

If the CFP duration is greater than the beacon interval, the PC shall transmit Beacon frames at the appropriate times during the CFP (subject to delay due to traffic at the nominal times, as with all Beacon frames). The CF Parameter Set element in all Beacon frames at the start of, or within, a CFP shall contain a non - zero value in the CFPDurRemaining field.

The PC may terminate any CFP at or before the aCFPMaxDuration, based on available traffic and size of the polling list. Because the transmission of any Beacon frame may be delayed due to a medium busy condition at the TBTT, a CFP may be foreshortened by the amount of the delay. In the case of a busy medium due to DCF traffic, the Beacon frame shall be delayed for the time required to complete the current DCF frame exchange.

At the nominal beginning of each CFP, the PC shall sense the medium. When the

medium is determined to be idle for one PIFS period, the PC shall transmit a Beacon frame containing the CF Parameter Set element and a DTIM element.

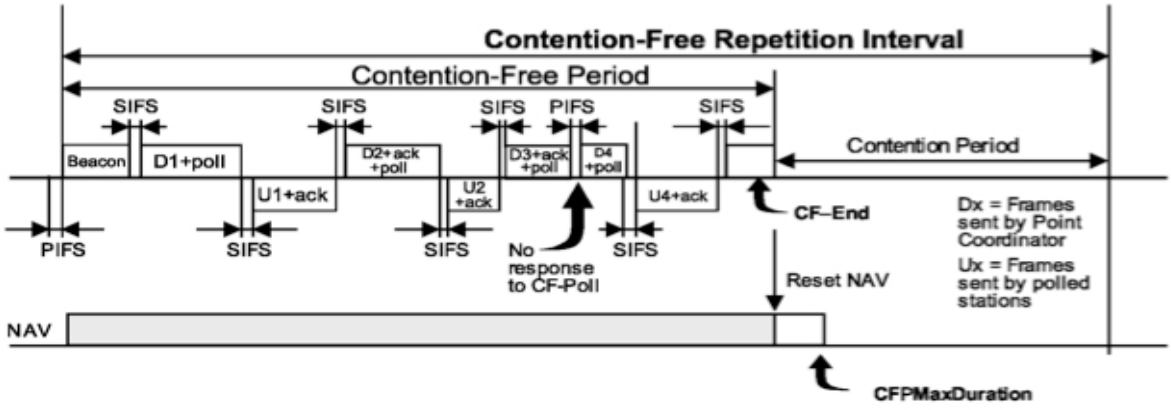


Figure 2.25 - PCF frame transfer

As illustrated on Figure 2.25, after the initial Beacon frame, the PC shall wait for one SIFS period, and then transmit one of the following: a data frame, a CF-Poll frame, a Data+CF-Poll frame, a management frame, or a CF-End frame. If the CFP is null, i.e., no traffic is buffered and no polls exist to send at the PC, a CF-End frame shall be transmitted immediately after the initial Beacon frame. If there are buffered multicast or broadcast frames, the PC shall transmit these prior to any unicast frames.

The PC shall transmit a CF-End or CF-End +ACK frame at the end of each CFP.

STAs receiving individually addressed, error-free frames from the PC are expected to respond after a SIFS period. If the recipient STA is not CF-Pollable, the response to receipt of an error-free data frame shall always be an ACK frame.

Each STA, except the STA with the PC, shall preset its NAV to the CFPMaxDuration value (obtained from the CF Parameter Set element in Beacon frames from this PC). This setting of the NAV also reduces the risk of hidden STAs determining the medium to be idle for a DIFS period during the CFP and possibly corrupting a transmission in progress.

Non-CF-Pollable STAs shall acknowledge receipt of data and management frames using ACK Control frames sent after a SIFS period. This non-CF- Pollable operation is the same as that already employed by such STAs for DCF operation.

When polled by the PCF (Data+CF-Poll, Data+CF-ACK+CF-Poll, CF-Poll, or CF-ACK+CF-Poll) a CF-Pollable STA may send one data frame to any destination. Such a frame directed to or through the PC STA shall be acknowledged by the PC, using the CF-ACK indication (Data+CF-ACK, Data+CF-ACK+CF-Poll,

If the PC supports use of the CFP for inbound frame transfer as well as for frame delivery, the PC shall maintain a "polling list" for use in selecting STAs that are eligible to receive CF-Polls during CFPs. If the PC supports the use of the CFP solely for frame delivery, the PC does not require a polling list, and shall never generate data frames with a subtype that includes CF-Poll. The form of CF support provided by the PC is identified in the Capability Information field of Beacon, Association Response, Reassociation Response, and Probe Response management frames, which are sent from APs. Any such frames sent by STAs, as in noninfrastructure networks, shall always have these bits set to 0.

The polling list is used to force the polling of CF-Pollable STAs, whether or not the PC has pending traffic to transmit to those STAs. The polling list may be used to control the use of Data+CF-Poll and Data+CF- ACK+CF-Poll types for transmission of data frames being sent to CF-Pollable STAs by the PC. The polling list is a *logical* construct, which is not exposed outside of the PC. A minimum set of polling list maintenance techniques are required to ensure interoperability of arbitrary CF-Pollable STAs in BSSs controlled by arbitrary APs with active PCs. APs may also implement additional polling list maintenance techniques that are outside the scope of this standard.

While time remains in the CFP, all CF frames have been delivered, and all STAs on the polling list have been polled, the PC *may* send data or management frames to *any* STAs.

A STA indicates its CF-Pollability using the CF-Pollable subfield of the Capability Information field of Association Request and Reassociation Request frames. If a STA desires to change the PC's record of CF- Pollability, that STA shall perform a re-association. During association, a CF-Pollable STA may request to be placed on the polling list, or to never be polled, by appropriate use of bits in the Capability Information field of the Associate Request or Reassociate Request frame, as shown in table 2.9.

2.5 Physical Layer (PHY) service specification.

Just like the IEEE 802.11 standard includes a common Medium Access Control

(MAC) Layer, which defines protocols that govern the operation of the wireless LAN; in addition, 802.11 comprises several alternative physical layers that specify the transmission and reception of 802.11 frames. Each PHY can consist of two protocols functions: **the Physical Layer Convergence Procedure (PLCP) and Physical Medium Dependent (PMD) sub-layers.** These are somewhat sophisticated terms that the standard uses to divide the major functions that occur within the Physical Layer. The PLCP prepares 802.11 frames for transmission and directs the PMD to actually transmit signals, change radio channels, receive signals, and so on.

The MAC layer communicates with the Physical Layer Convergence Protocol (PLCP) sublayer via primitives (a set of "instructive commands" or "fundamental instructions") through a service access point (SAP). When the MAC layer instructs it to do so, the PLCP prepares MAC protocol data units (MPDUs) for transmission. The PLCP minimizes the dependence of the MAC layer on the PMD sublayer by mapping MPDUs into a frame format suitable for transmission by the PMD. The PLCP also delivers incoming frames from the wireless medium to the MAC layer.

The PLCP appends a PHY-specific preamble and header fields to the MPDU that contain information needed by the Physical layer transmitters and receivers (Figure 2.26). The 802.11 standard refers to this composite frame (the MPDU with an additional PLCP preamble and header) as a PLCP protocol data unit (PPDU). The MPDU is also called the PLCP Service Data Unit (PSDU), and is typically referred to as such when referencing physical layer operations.

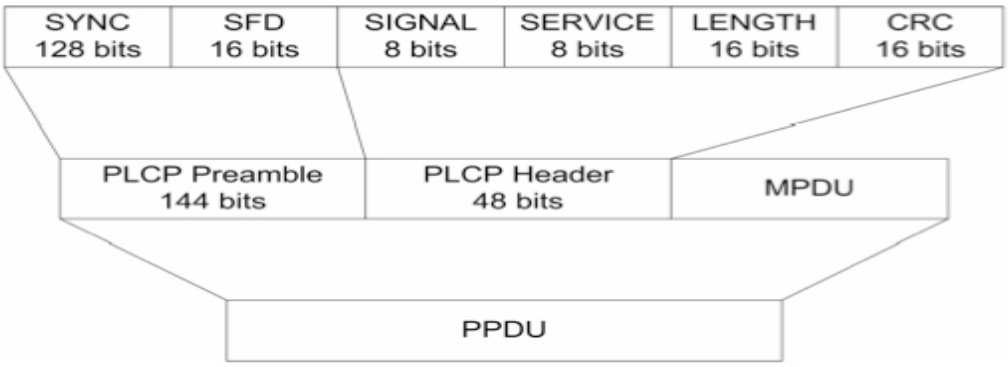


Figure 2.26 - PLCP frame format

What follows is a description of the fields that compose the PLCP preamble and the PLCD header:

- **Sync**. This field consists of alternating 0s and 1s, alerting the receiver that a receivable signal is present. The receiver begins synchronizing with the incoming signal after detecting the Sync.

- **Start Frame Delimiter**. This field is always 1111001110100000 and defines the beginning of a frame.

- **Signal**. This field identifies the data rate of the 802.11 frame, with its binary value equal to the data rate divided by 100Kbps. For example, the field contains the value of 00001010 for 1Mbps, 00010100 for 2Mbps, and so on. The PLCP fields, however, are always sent at the lowest rate, which is 1Mbps. This ensures that the receiver is initially uses the correct demodulation mechanism, which changes with different data rates.

- **Service**. This field is always set to 00000000, and the 802.11 standard reserves it for future use.

- **Length**. This field represents the number of microseconds that it takes to transmit the contents of the PPDU, and the receiver uses this information to determine the end of the frame.

- **Frame Check Sequence**. In order to detect possible errors in the Physical Layer header, the standard defines this field for containing 16-bit cyclic redundancy check (CRC) result. The MAC Layer also performs error detection functions on the PPDU contents as well.

PSDU. The PSDU, which stands for Physical Layer Service Data Unit, is a fancy name that represents the contents of the PPDU .

Under the direction of the PLCP, the Physical Medium Dependent (PMD) sub-layer provides transmission and reception of Physical layer data units between two stations via the wireless medium. To provide this service, the PMD interfaces directly with the wireless medium (that is, RF in the air) and provides modulation and demodulation of the frame transmissions. The PLCP and PMD sub-layers communicate via primitives, through

a SAP[8], to govern the transmission and reception functions. This interaction is illustrated in Figure 2.27.

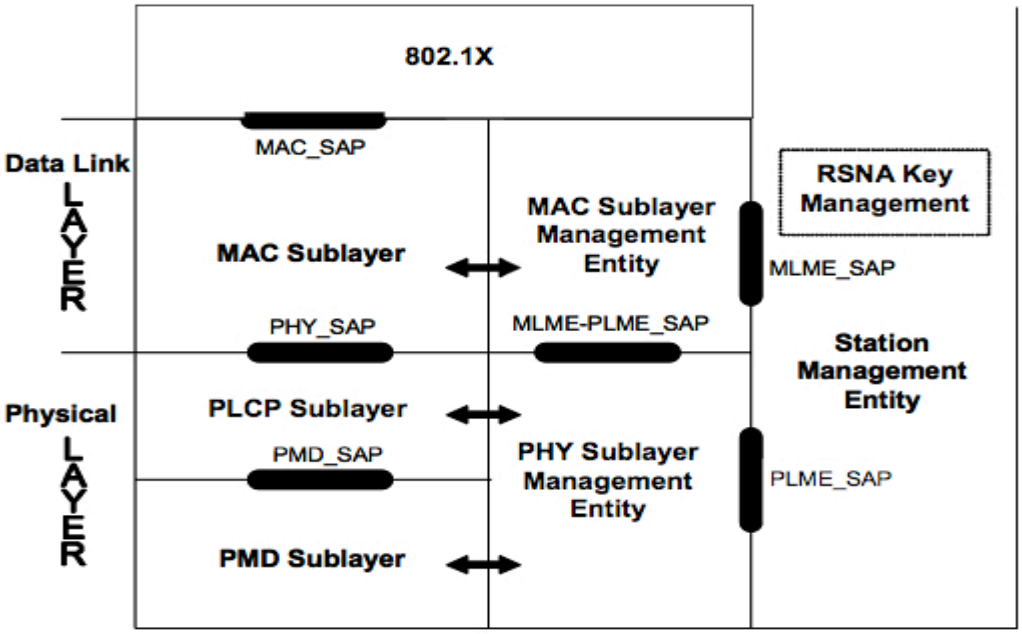


Figure 2.27 - Portion of the ISO/IEC basic reference model

The general operation of the various Physical layers is very similar. To perform PLCP functions, the 802.11 standard specifies the use of state machines. Each state machine performs one of the following functions:

1. **Carrier Sense/Clear Channel Assessment (CS/CCA)**

Carrier Sense/Clear Channel Assessment is used to determine the state of the medium. The CS/CCA procedure is executed while the receiver is turned on and the station is not currently receiving or transmitting a packet. The CS/CCA procedure is used for two specific purposes: to detect the start of a network signal that can be received (CS) and to determine whether the channel is clear prior to transmitting a packet (CCA).

2. **Transmit (Tx)**

Transmit (Tx) is used to send individual octets of the data frame. The transmit

[8] Service Access Point

procedure is invoked by the CS/CCA procedure immediately upon receiving a PHY-TXSTART.request (TXVECTOR) from the MAC sublayer. The CSMA/CA protocol is performed by the MAC with the PHY PLCP in the CS/CCA procedure prior to executing the transmit procedure.

3. Receive (Rx)

Receive (Rx) is used to receive individual octets of the data frame. The receive procedure is invoked by the PLCP CS/CCA procedure upon detecting a portion of the preamble sync pattern followed by a valid SFD and PLCP Header. Although counter-intuitive, the preamble and PLCP header are not "received". Only the MAC frame is "received".

Following, a list of formats used for PSDU transmission:

The **IEEE 802.11b Direct Sequence Spread Spectrum (DSSS) Physical layer** (802.11b) delivers frames at 1, 2, 5.5, and 11 Mbps rates in the 2.4 GHz ISM band. The original 802.11 Clause 15 DSSS standard specified only 1 and 2 Mbps data rates using only long preambles. The only coding/modulation used in 802.11 is Barker code with DBPSK (1 Mbps) and DQPSK (2 Mbps). Figure 2.28 illustrates the construction of the DSSS PPDU, which includes a long preamble, the header, and the MPDU (PSDU) as specified in the 802.11 standard. The preamble and the header are both transmitted at 1 Mbps when using the long preamble format. The MPDU is transmitted at the data rate specified by the transmitting station (or access point). The preamble enables the receiver to synchronize to the incoming signal properly before the actual content of the frame arrives. The header provides information about the frame, and the PSDU is the MPDU the transmitting station is sending.

The 802.11b standard further specifies rates of 5.5 and 11 Mbps, each using CCK modulation.

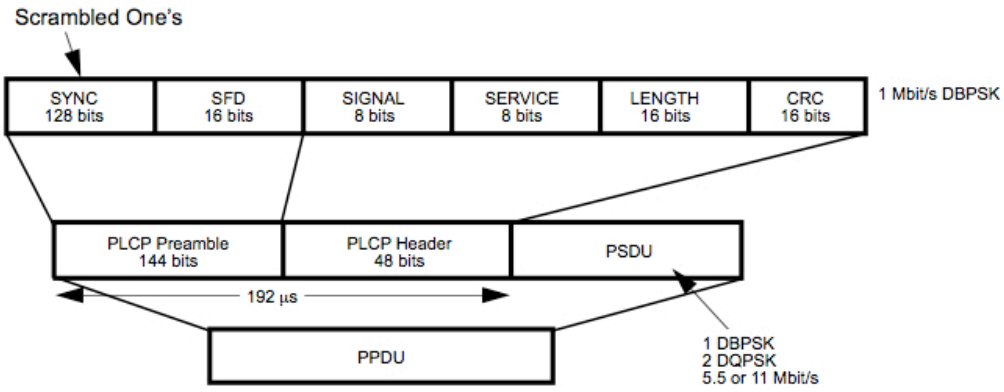


Figure 2.28 - Long preamble version's format

The option of a short preamble was introduced in 1997, as an optional alternative, giving the administrator two configuration options. This is an illustration of the short version:

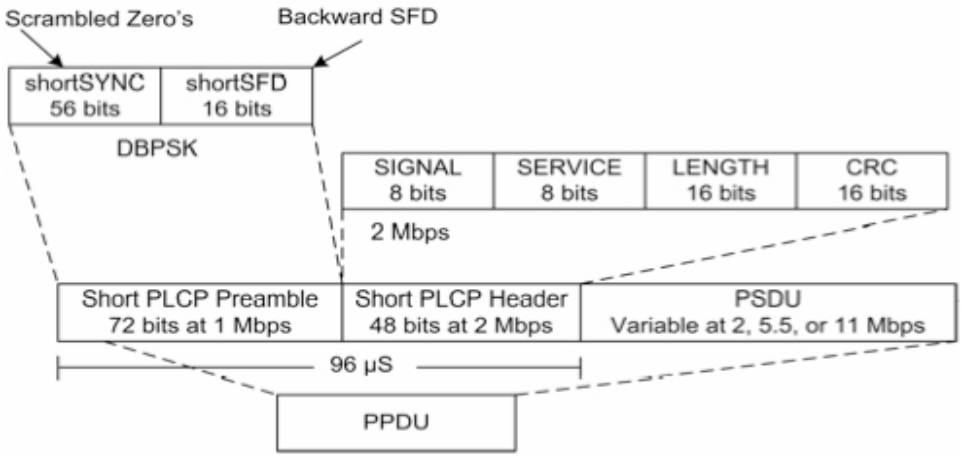


Figure 2.29 - Short preamble version's format

Other PHY specifications are illustrated in next table:

Characteristic	Value
aSlotTime	20 μs
aSIFSTime	10 μs
aCCATime	≤ 15 μs
aPHY-RX-START-Delay	192 μs
aRxTxTurnaroundTime	≤ 5 μs
aTxPLCPDelay	Implementers may choose any value for this delay as long as the requirements of aRxTxTurnaroundTime are met.
aRxPLCPDelay	Implementers may choose any value for this delay as long as the requirements of aSIFSTime and aCCATime are met.
aRxTxSwitchTime	≤ 5 μs
aTxRampOnTime	Implementers may choose any value for this delay as long as the requirements of aRxTxTurnaroundTime are met.
aTxRampOffTime	Implementers may choose any value for this delay as long as the requirements of aSIFSTime are met.
aTxRFDelay	Implementers may choose any value for this delay as long as the requirements of aRxTxTurnaroundTime are met.
aRxRFDelay	Implementers may choose any value for this delay as long as the requirements of aSIFSTime and aCCATime are met.
aAirPropagationTime	1 μs
aMACProcessingDelay	≤ 2 μs
aPreambleLength	144 μs
aPLCPHeaderLength	48 μs
aMPDUDurationFactor	0
aMPDUMaxLength	$4 \leq x \leq (2^{13} - 1)$
aCWmin	31
aCWmax	1023

Table 2.14 - High Rate PHY characteristics

The OFDM system (IEEE 802.11a) provides a WLAN with data payload communication capabilities of 6, 9, 12, 18, 24, 36, 48, and 54 Mb/s. The support of transmitting and receiving at data rates of 6, 12, and 24 Mb/s is mandatory. The system uses 52 subcarriers that are modulated using binary or quadrature phase shift keying (BPSK or QPSK) or using 16- or 64-quadrature amplitude modulation (16-QAM or 64-QAM). Forward error correction coding (convolutional coding) is used with a coding rate of 1/2, 2/3, or 3/4.

The support of transmitting and receiving at data rates of 3, 6, and 12 Mb/s is mandatory when using 10 MHz channel spacing. The half- clocked operation doubles symbol times and clear channel assessment (CCA) times when using 10 MHz channel spacing.

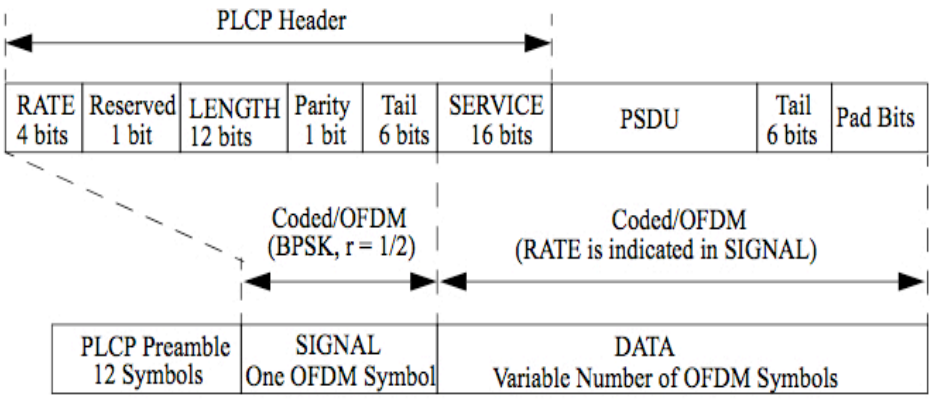


Figure 2.30 - OFDM P LCP frame format

Another PHY specifications are illustrated in the table 2.15:

Characteristics	Value (20 MHz channel spacing)	Value (10 MHz channel spacing)	Value (5 MHz channel spacing)
aSlotTime	9 µs	13 µs	21 µs
aSIFSTime	16 µs	32 µs	64 µs
aCCATime	< 4 µs	< 8 µs	< 16 µs
aPHY-RX-START-Delay	25 µs	49 µs	97 µs
aRxTxTurnaroundTime	< 2 µs	< 2 µs	< 2 µs
aTxPLCPDelay	Implementation dependent as long as the requirements of aRxTxTurnaroundTime are met.	Implementation dependent as long as the requirements of aRxTxTurnaroundTime are met.	Implementation dependent as long as the requirements of aRxTxTurnaroundTime are met.
aRxPLCPDelay	Implementation dependent as long as the requirements of aSIFSTime and aCCA-Time are met.	Implementation dependent as long as the requirements of aSIFSTime and aCCA-Time are met.	Implementation dependent as long as the requirements of aSIFSTime and aCCA-Time are met.
aRxTxSwitchTime	<< 1 µs	<< 1 µs	<< 1 µs
aTxRampOnTime	Implementation dependent as long as the requirements of aRxTxTurnaroundTime are met.	Implementation dependent as long as the requirements of aRxTxTurnaroundTime are met.	Implementation dependent as long as the requirements of aRxTxTurnaroundTime are met.
aTxRampOffTime	Implementation dependent as long as the requirements of aSIFSTime are met.	Implementation dependent as long as the requirements of aSIFSTime are met.	Implementation dependent as long as the requirements of aSIFSTime are met.
aTxRFDelay	Implementation dependent as long as the requirements of aRxTxTurnaroundTime are met.	Implementation dependent as long as the requirements of aRxTxTurnaroundTime are met.	Implementation dependent as long as the requirements of aRxTxTurnaroundTime are met.
aRxRFDelay	Implementation dependent as long as the requirements of aSIFSTime and aCCA-Time are met.	Implementation dependent as long as the requirements of aSIFSTime and aCCA-Time are met.	Implementation dependent as long as the requirements of aSIFSTime and aCCA-Time are met.
aAirPropagationTime	<< 1 µs	<< 1 µs	<< 1 µs
aMACProcessingDelay	< 2 µs	< 2 µs	< 2 µs
aPreambleLength	16 µs	32 µs	64 µs
aPLCPHeaderLength	4 µs	8 µs	16 µs
aMPDUMaxLength	4095	4095	4095
aCWmin	15	15	15
aCWmax	1023	1023	1023

Figure 2.15 - OFDM PHY characteristics

The Extended Rate PHY (IEEE 802.11g) works in the 2.4 GHz band (like 802.11b), but uses the same OFDM based transmission scheme as 802.11a. It operates at a

maximum physical layer bit rate of 54 Mbit/s exclusive of forward error correction codes, or about 22 Mbit/s average throughputs.

The ERP builds on the payload data rates of 1 and 2 Mb/s, as described in (802.11 b)that use DSSS modulation and builds on the payload data rates of 1, 2, 5.5, and 11 Mb/s, as described(802.11 b) that use DSSS, CCK, and optional PBCC modulations. The ERP draws to provide additional payload data rates of 6, 9, 12, 18, 24, 36, 48, and 54 Mb/s. Of these rates, transmission and reception capability for 1, 2, 5.5, 6, 11, 12, and 24 Mb/s data rates is mandatory.

Two additional optional ERP-PBCC modulation modes with payload data rates of 22 and 33 Mb/s are defined. An ERP-PBCC STA may implement 22 Mb/s alone or 22 and 33 Mb/s. An optional modulation mode known as DSSS-OFDM is also incorporated with payload data rates of 6, 9, 12, 18, 24, 36, 48, and 54 Mb/s.

An ERP STA shall support three different preamble and header formats. The first is the Long Preamble and header described in Figure 2.28.This PPDU provides interoperability with IEEE 802.11b STAs when using the 1, 2, 5.5, and 11 Mbit/s data rates; the optional DSSS-OFDM modulation at all OFDM rates; and the optional ERP-PBCC modulation at all ERP- PBCC rates. The second is the Short Preamble and header described in 2.29. The short preamble supports the rates 2, 5.5, and 11 Mbit/s as well as DSSS-OFDM and ERP-PBCC. The third is the ERP-OFDM preamble and header. In the case of modulations ERP-DSSS, ERP-CCK, DSSS-OFDM and ERP_PBCC, the utilized format can be short or long preamble, as shown in figures 2.28 and 2.29. However, under ERP-OFDM modulation on a physical level, the format corresponds to figure 2.30.

Another PHY specifications about this ERP version are illustrated next:

Characteristic	Value
aSlotTime	Long = 20 µs, short = 9 µs
aSIFSTime	10 µs
aCCATime	<15 µs for long slot time or <4 µs for Short Slot Time, see 19.4.6
aRxTxTurnaroundTime	<5 µs
aTxRxTurnaroundTime	<10 µs
aTxPLCPDelay	Implementation dependent as long as the requirements of aRxTxTurnaround-Time are met.
aRxPLCPDelay	Implementation dependent as long as the requirements of aSIFSTime and aCCATime are met.
aRxTxSwitchTime	<<1 µs
aTxRampOnTime	Implementation dependent as long as the requirements of aRxTxTurnaround-Time are met.
aTxRampOffTime	Implementation dependent as long as the requirements of aSIFSTime are met.
ATxRFDelay	Implementation dependent as long as the requirements of aRxTxTurnaround-Time are met.
ARxRFDelay	Implementation dependent as long as the requirements of aSIFSTime and aCCATime are met.
aAirPropagationTime	<<1 µs
aMACProcessingDelay	<2 µs
aPreambleLength	20 µs
aPLCPHeaderLength	4 µs
aMPDUMaxLength	4095
aCWmin(0)	31

Table 2.16 - ERP PHY characteristics

One important aspect is the list of WLAN channels, is the legally allowed IEEE 802.11 or more commonly Wi-Fi Wireless LAN channels.

The 802.11 workgroup currently documents use in three distinct frequency ranges, 2.4 GHz, 3.6 GHz and 4.9/5.0 GHz bands [4]. Each range is divided into multitude of channels. Countries apply their own regulations to both the allowable channels, allowed users and maximum power levels within these frequency ranges.

These regulations are subject to change at any time.

About 2.4 GHz band, potential Wireless LAN uses this range (802.11b, 802.11g and 802.11n). In this band are 14 channel spaced 5 MHz apart (with the exception of a 12 MHz spacing before Channel 14). As the protocol requires 25 MHz of channel separation, adjacent channels overlap and will interfere with each other potential uses of this range.

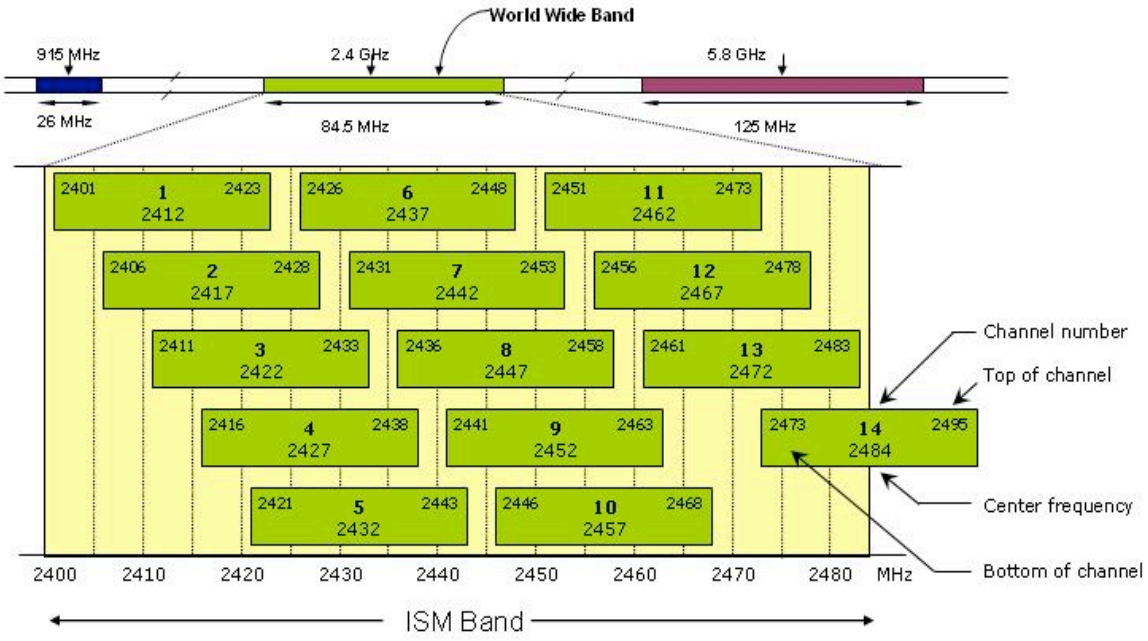


Figure 2.31 - 2.4 GHz band

The 3.6 GHz band range is documented as only being allowed as a licensed band in the United States.

The 5 GHz frequency band offers at least 19 non-overlapping channels rather than the 3 offered in the 2.4 GHz band. Better or worse performance with higher or lower frequencies (channels) may be realized, depending on the environment.

Chapter 3

PATTERN RECOGNITION BY LINEAR CLASSIFICATION APPROACH

3.1 Pattern Recognition task

Pattern recognition is the scientific discipline whose goal is the classification of objects into a number of categories or classes.

"The ease with which we recognize a face, computer-aided diagnosis,understand spoken words , read handwritten characters, identify our car keys in our pocket by feel, and decide whether an apple is ripe by its smell belies the astoundingly complex processes that underlie these acts of pattern recognition". [5]

The foregoing are just some examples from a much larger number of possible applications. Of course, to achieve the final goals in all of the applications, pattern recognition is closely linked with other scientific disciplines, such as linguistics, computer graphics, machine vision, and database design.

Figure 3.1 shows the various stages of a complete pattern recognition system. These consists of a sensor that gathers the observations to be classified or described, a feature generation mechanism that computes numeric or symbolic information from the observations, a feature selection block (selecting which type of characteristics are more adequate to describe an object) and a classification scheme that does the actual job of classifying or describing

observations, relying on the extracted features. The task of the system evaluation stage is to assess the classifier's performance.

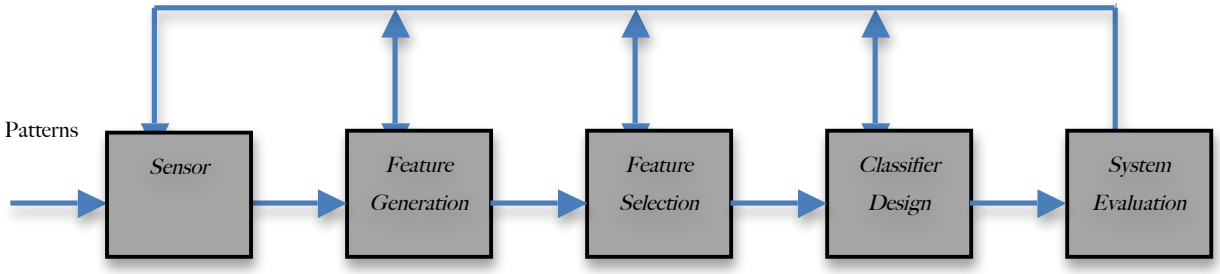


Figure 3.1 - Basic stages envolved in the design of a classification system

The classification design is usually based on the availability of a set of patterns that have already been classified or described. This set of patterns is termed the ***training set***, and the resulting learning strategy is characterized as supervised learning. Learning can also be unsupervised, in the sense that the system is not given an *a priori* labeling of patterns, instead it itself establishes the classes based on the statistical regularities of the patterns.

The classification or description scheme usually uses one of the following approaches: statistical (also known as decision theoretic) or syntactic (also known as structural). Statistical pattern recognition is based on statistical characterizations of patterns, assuming that the patterns are generated by a probabilistic system. Syntactical pattern recognition is based on the structural interrelationships of features.

We will focus on the design of **linear classifiers,** regardless of the underlying distributions describing the training data. The major advantage of linear classifiers is their simplicity and computational attractiveness. A **linear classifier** basically works by making a classification decision based on the value of a linear combination of the features.

3.2 Linear Discrimination Functions and Decision Hyperplanes

A discriminant function [5] that is a linear combination of the components of **X** can be written as:

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i, \tag{3.1}$$

Where w= [$w1$,$w2$,……..,wd] is know as the weight vector and w_0 as the threshold .And X= [$x1$,$x2$,… xd] is a point on the decision hyperplane.

A two-category linear classifier implements the following decision rule: decide ω1 if $g(x) > 0$ and ω2 if $g(x) < 0$. Thus, x is assigned to ω1 if the inner product $\boldsymbol{w}^T\mathbf{X}$ exceeds the threshold $-w_0$ and ω2 otherwise. . Figure 3.2 displays a structural design, explaining the discriminant function implementation.

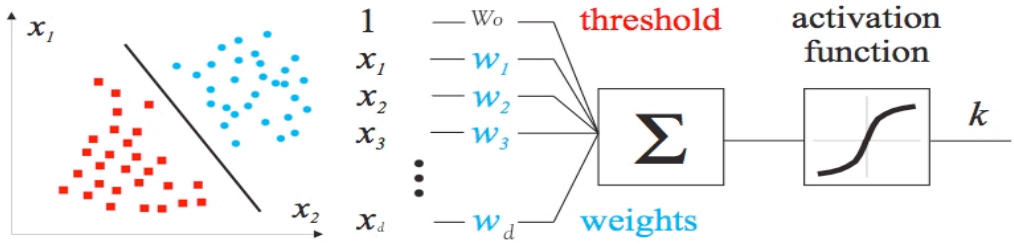


Figure 3.2 - An illustration of two-class case and decision rule

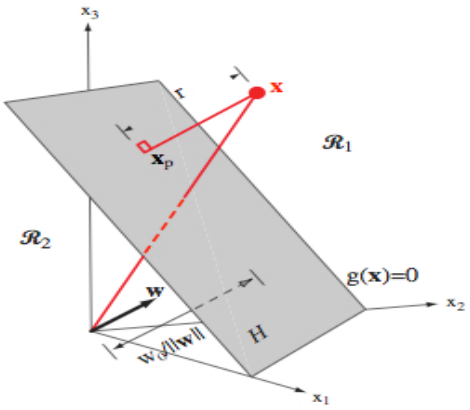
If g(x)=0, **X** can ordinarily be assigned to either class. The equation g(x)=0 defines the decision surface that separates points assigned to ω1 from points assigned to ω2. When g(x) is linear, this decision surface is a hyperplane. If **X1** and **X2** are both on the decision surface, then:

$$\mathbf{w}^t\mathbf{x}_1 + w_0 = \mathbf{w}^t\mathbf{x}_2 + w_0$$

$$\mathbf{w}^t(\mathbf{x}_1 - \mathbf{x}_2) = 0, \tag{3.2}$$

Since the difference vector ***X1- X2*** obviously lies on the decision hyperplane (for any ***X1, X2***), it apparent from Eq.(3.2) that the vector w is orthogonal to the decision hyperplane.

The figure 3.3 shows the corresponding geometry in two-class case:



Figue 3.3 - The linear decision boundary

The linear decision boundary H, where $g(x) = \boldsymbol{w}^T\boldsymbol{X} + w_0 = 0$, separates the feature space into two half-spaces R1 (where $g(x) > 0$) and R2 (where $g(x) < 0$).

The discriminant function $g(x)$ gives an algebraic measure of the distance from X to the hyperplane. Perhaps the easiest way to see this is to express X as:

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}, \tag{3.3}$$

where Xp is the normal projection of X onto H, and r is the desired algebraic distance positive if X is on the positive side and negative if X is on the negative side. Then, since $g(Xp) = 0$,

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = r\|\mathbf{w}\|,$$

(3.4)

or

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}.$$

(3.5)

In particular, the distance from the origin to H is given by $/\|w\|$. If $w_0 > o$ the originis on the positive side of H, and if $w_0 < o$ it is on the negative side. If $w_0 = o$, then g(x) has the homogeneous form $\boldsymbol{w^TX}$, and the hyperplane passes through the origin.

To summarize, a linear discriminant function divides the feature space by a hyperplane decision surface. The orientation of the surface is determined by the normal vector w and the location of the surface is determined by the bias w_0. The discriminant function $g(x)$ is proportional to the signed distance from X to the hyperplane, with $g(x) > o$ when **X** is on the positive side, and $g(x) < o$ when x is on the negative side.

Our major concern now is to compute the unknown parameters w= [$w1,w2,\ldots\ldots,wd$] defining the decision hyperplanes. A great variety of linear classification algorithms is available depending on whether or not these or similar conditions are verified: separability of the classes to classify, knowledge of statistical information that identifies each class or the context in which the classification is made, as well as others. However, the vast majority of these algorithms follow one of the following methods of linear classification:

- Perceptron Algorithm .One of the oldest algorithms used in machine learning. A basic requirement for the convergence is the linear separability of the classes. With perceptron mechanism we will approach the problem as a typical optimization task . Thus we need to adopt an appropriate cost function and an algorithmic scheme to optimize it. To this end, we choose the perceptron cost [5] defined as:

$$J(w) = \sum_{x \in Y} \delta x w^T X \quad , \qquad (3.6)$$

where Y is the subset of the training vectors, which are misclassified by the hyper-

plane defined by the weight vector w. The variable δx, is chosen so that δx=- 1 if $\mathbf{X} \in \omega_1$ and δx= 1 if $\mathbf{X} \in \omega_2$. Obviously, the sum in (3.6) is always positive and it becomes zero when Y becomes the empty set, that is, if there are not misclassified vectors $\mathbf{X}$. Indeed, if $\mathbf{X} \in \omega_1$ and it is misclassified, then $w^T X + w_0 < 0$ and δx < 0, and the product is positive. The result is the same for vectors originating from class ω2. When the cost function takes its minimum value, 0, a solution has been obtained, since all training feature vectors are correctly classified.

To derive the algorithm for the iterative minimization of the cost function, we will adopt an iterative scheme in the spirit of the gradient descent method, that is

$$w(t+1) = w(t) - \rho_t \frac{\partial J(w)}{\partial w} \Big|_{w=w(t)} \tag{3.7}$$

where $w(t)$ is the weight vector estimate at the t-th iteration step, and ρt is a sequence of positive real numbers. However, we must be careful here. This is not defined at the points of discontinuity. From the definition in (3.6), and at the points where this is valid, we get:

$$\frac{\partial J(w)}{\partial w} = \sum_{X \in Y} \delta_X X \tag{3.8}$$

Substituting (3.8) into (3.7) we obtain:

$$w(t+1) = w(t) - \rho_t \sum_{X \in Y} \delta_X X \tag{3.9}$$

The algorithm is initialized from an arbitrary weight vector $\mathbf{w(0)}$, and the correction vector $\Sigma \delta_x \mathbf{X}$ is formed using the misclassified features. The weight vector is then corrected according to the preceding rule. This is repeated until the algorithm converges to a solution, that is, all features are correctly classified.

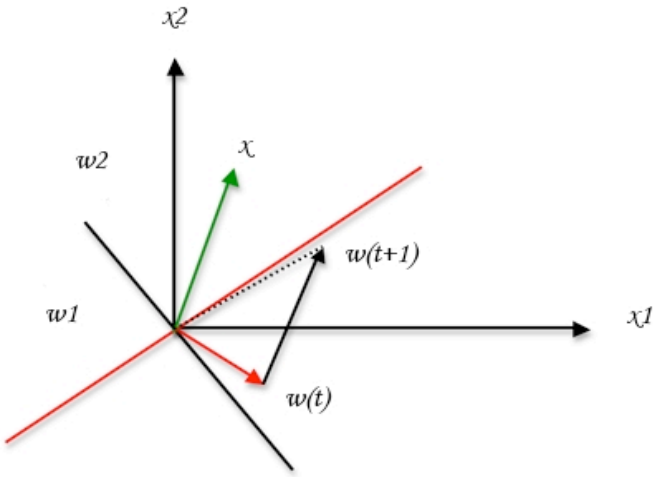


Figure 3.4 - Geometric interpretation of the perceptron algorithm

❖ Least Squares Methods. As we have already pointed out, the attractiveness of linear classifiers lies in their simplicity. Thus, in many cases, although we know that the classes are not linearly separable, we still wish to adopt a linear classifier, despite the fact that this will lead to suboptimal performance from the classification error probability point of view. The least squares methods goal is to compute the corresponding weight vector under a suitable optimality criterion [5].

In this case, the weight vector will be computed so as minimize the mean square error (MSE) between the desired and true outputs, that is:

$$J(w) = E\left[\left|y - X^T w\right|^2\right] \quad , \qquad (3.10)$$

$$w = arg\ min\ J(w) \qquad (3.11)$$

❖ Mean Square Estimation Revisited. Let $\boldsymbol{y}$, $\mathbf{X}$ be two random vectors of dimensions ($\mathbf{M}$ x $\mathbf{1}$) and (l x 1), respectively, and assume that they are described by the joint pdf [1] $\boldsymbol{p}(\mathbf{y},\mathbf{X})$. The task of interest is to estimate the value of y, given the value of $\mathbf{X}$, obtained from an experiment. [5] No doubt the classification task falls under this more general formulation.

❖ Support Vector Machines. More formally, a support vector machine constructs a hyperplane or set of hyperplanes [5] in a high or infinite dimensional space, which could be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training datapoints of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

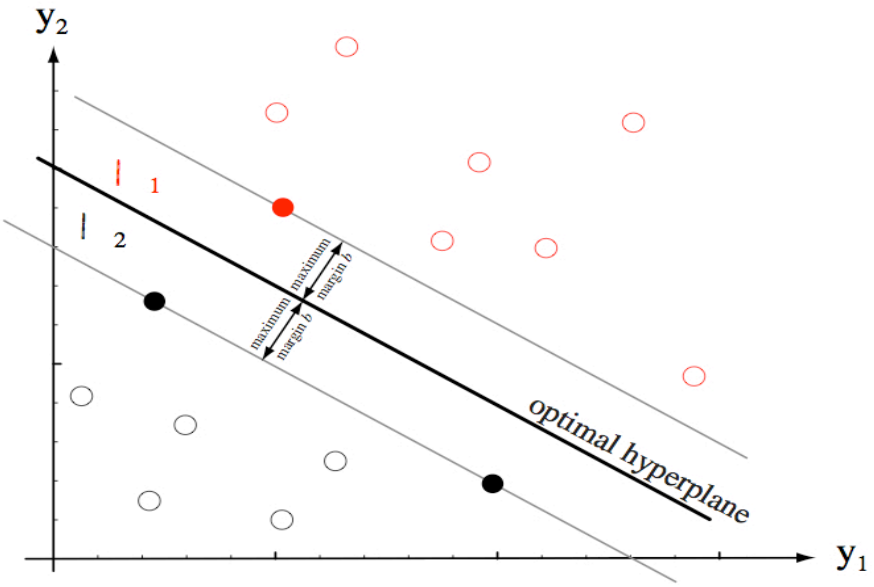


Figure 3.5 - Geometric interpretation of the Support Vector Machines

3.3 The Pocket Algorithm

The pocket algorithm is a modification of perceptron learning that makes perceptron learning well behaved with non separable training data [6], even if that data is noisy and

[1] Probability Density Function

contradictory. Features of these algorithms include: speed -fast enough to be able to handle large sets of training data- and scaling properties -when the number of inputs is increases-.

Non separable problems are a different story. Since no set of weights can correctly classify all training examples, the best that can be hoped for is a set of weights that correctly classifies as large a fraction of the training examples as possible. Such a set of weights is called optimal.

Note that there are alternatives that do not fit the training data as well, for example computing weights that give minimum squared error. Such alternatives are necessary for algorithms, such as back-propagation, that require a differentiable error function.

Perceptron learning is not well behaved for non separable problems. While it will eventually visit an optimal set of weights, it will not converge to any set of weights. Even worse, the algorithm can go from an optimal set of weights to a worst-possible set in one iteration, regardless of how many iterations have been taken previously. The pocket algorithm makes perceptron learning well behaved by adding positive feedback in order to stabilize the algorithm.

There are several variants for different classes of problems. We are focused on the ***Pocket algorithm with Ratchet.***

Applicability: Finite set of training examples. Examples may be repeated, noisy, and contradictory ($E_k = E_l$, $C_k \neq C_l$).

Algorithm: The basic idea of perceptron learning is to take a training example E_k, that is incorrectly classified by the current set of weights and to add E_k to the current weights if $C_k = 1$ or subtract E_k from the current weights if $C_k = -1$.

The basic idea of the pocket algorithm is to run perceptron learning while keeping an extra set of weights "in your pocket." Whenever the perceptron weights have a longest **run** of consecutive correct classifications of randomly selected training examples, these perceptron weights replace the pocket weights.

3.4 Stochastic Approximation and LMS

The solution of (3.10) requires the computation of the correlation matrix and cross-correlation vector. This presupposes knowledge of the underlying distributions, which in

general are not known. Thus, our major goal now becomes to see if it is possible to solve (3.11) without having this statistical information available. Consider an equation of the form **E[F(*Xk,w)*]=*0*, where ***Xk,*** k=1,2,...,is a sequence of random vectors from the same distribution, $F(\,.\,,.\,)$ a function, and $\boldsymbol{w}$ the vector of the unknown parameters. Then adopt the iterative scheme [7]:

$$w(k) = w(k-1) - \rho_k F(X_k, w(k-1)) \tag{3.12}$$

In other words, the place of the mean value (which cannot be computed due to lack of information) is taken by the samples of the random variables resulting from the experiments. It turns out that under mild conditions the iterative scheme converges in probability to the solution $\boldsymbol{w}$ of the original equation, provided that the sequence ρ_k satisfies the two conditions:

$$\sum_{k=1}^{\infty} \rho_k \to \infty$$
$$\sum_{k=1}^{\infty} \rho_k{}^2 < \infty \tag{3.13}$$

Let us now return to our original problem and apply the iteration to solve (3.11) the (3.12) becomes:

$$w(k) = w(k-1) + \rho_k X_k (y_k - X_k{}^T w(k-1)) \tag{3.14}$$

where (y_k, X_k) are the desired output(±1) input training sample pairs, successively presented to the algorithm. The algorithm is know as the least mean squares (LMS) or Widrow_Hoff algorithm. The algorithm converges asymptotically to the MSE solution.

3.5 Sum of Error Squares Estimation

A criterion closely related to the MSE is the sum of error squares criterion defined as [7]:

$$J(w) = \sum_{k=1}^{N} (y_k - X_k^T w)^2 = \sum_{k=1}^{N} e_k^2 \tag{3.15}$$

In other words, the errors between the desired output of the classifier (±1 in the two class case) and the true output are summed up over all the available training feature vectors, instead of averaging them out. In this way we overcome the need for explicit knowledge of the underlying pdf's. Minimizing (3.15)with respect to w results in:

$$\sum_{k=1}^{N} X_k (y_k - X_k^T \hat{w}) = 0 \Rightarrow (\sum_{k=1}^{N} X_k X_k^T) \hat{w} = \sum_{k=1}^{N} (X_k y_k) \tag{3.16}$$

For the sake of mathematical formulation, let us define:

$$Q = \begin{pmatrix} X_1^T \\ X_2^T \\ . \\ . \\ X_N^T \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} & X_{13} & \ldots & X_{1l} \\ X_{21} & X_{22} & X_{23} & \ldots & X_{2l} \\ X_{31} & X_{32} & X_{33} & \ldots & X_{3l} \\ . & . & . & . & . \\ X_{N1} & X_{N2} & X_{N3} & \ldots & X_{Nl} \end{pmatrix} \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{pmatrix} \tag{3.17}$$

That is, Q is an *(N x l)* matrix whose rows are the available training feature vectors, and y is a vector consisting of the corresponding desired responses. Then:

$$\begin{aligned} \sum_{i=1}^{N} X_i X_i^T &= QQ^T \\ \sum_{i=1}^{N} X_i y &= Q^T y \end{aligned} \tag{3.18}$$

Hence, Equation (3.18) can now be written as:

$$(Q^T Q) \hat{w} = Q^T y \Rightarrow \hat{w} = (Q^T Q)^{-1} Q^T y \tag{3.19}$$

Thus, the optimal weight vector is again provided as the solution of a linear set of equations. Matrix Q^TQ is known as the sample correlation matrix. Matrix $Q^+ = (Q^TQ)^{-1}Q^T$ is known as the pseudoinverse of Q, and it is meaningful only if Q^TQ is invertible, that is, Q is of rank l. If Q is an (l×l) square and invertible matrix, then it is straightforward to see that $Q^+ = Q^{-1}$ .In such a case the estimated weight vector is the solution of the linear system Xw = y. If, however, there are more equations than unknowns, $N > l$, as is the usual case in pattern recognition. there is not, in general, a solution. The solution obtained by the pseudoinverse is the vector that minimizes the sum of error squares. It is easy to show that (under mild assumptions) the sum of error squares tends to the MSE solution for large values of N.

3.6 Logistic Discrimination

In logistic discrimination [7] the logarithm of the likelihood ratios is modeled via linear function. That is,

$$\ln \frac{P(\omega_i|X)}{P(\omega_M|X)} = w_{i,0} + w_i^T X \tag{3.20}$$

In the denominator, any class other than ω_M can also be used. The unknown parameters wi, must be chosen to ensure that probabilities add to one. That ,is:

$$\sum P(\omega_i X|) = 1 \tag{3.21}$$

Combining (3.20) and (3.21), it is straightforward to see that this type of linear modeling is equivalent to an exponential modeling of the a posteriori probabilities:

$$P(\omega_M|X) = \frac{1}{1+\sum \exp(w_{i,0} + w_i^T X)} \qquad i = 1,2,...,M-1 \tag{3.22}$$

$$P(\omega_i|X) = \frac{\exp(w_{i,0} + w_i^T X)}{1 + \sum \exp(w_{i,0} + w_i^T X)} \qquad i = 1,2,...,M-1 \qquad (3.23)$$

To estimate the set of the unknown parameters, a maximum likelihood approach is usually employed.

3.7 Multi-Class Case

It often becomes necessary to classify by differentiation amongst more than two classes. Generalization in this cases turns into a complex matter. A linear discriminator function is defined for each one of the classes, taking the following form: ω_i, i=1,2,...M. A feature vector (in the(l+1)-dimensional space to account for the threshold) is classified in ω_i class if:

$$w_i^T X > w_j^T X \qquad \forall j \neq i \qquad (3.24)$$

Here we describe plausible strategies to classify in such cases [5]:

✓ Building M linear classifiers by solving ω_i, /not ω_i, dichotomies (Linear Machine)

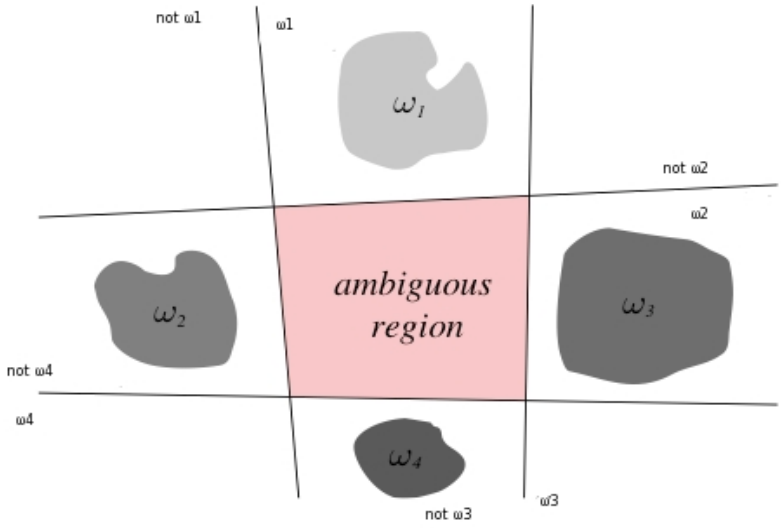


Figure 3.6 - Linear decision boundaries for a four-class problem. ωi/not ωi dichotomies

- ✓ Use $M(M-1)/2$ perceptrons: one for every pair ω_i/ω_j, dichotomies

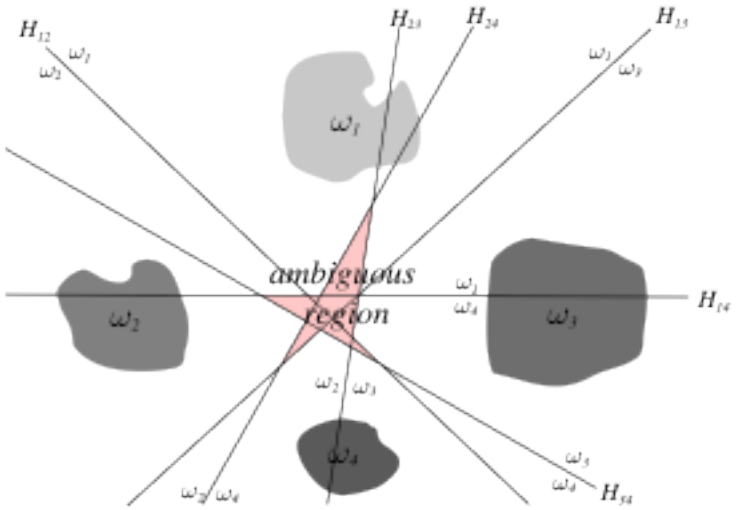


Figure 3.7 - Linear decision boundaries for a four-class problem. ωi /ωj dichotomies and the corresponding decision boundaries H_{ij}.

Regardless of the classification algorithm in use, a linear machine can be implemented using *c* linear neurons, one for each category and each one with the corresponding decision hyperplane $g_i(X)$. Assigning X to ùi if :

$$g_i(X) > g_j(X) \quad \forall j \neq I \tag{3.25}$$

A linear machine divides the feature space into M decision regions, with $g_i(X)$ being the largest discriminant if X is in region *Ri.*

We can transform the linear machine learning problem to that of learning a single perceptron by using the Kesler's construction [7].

In this case, we will perform an expansion of feature's space, following these steps:

1. For each training vector of the ωi class, *(M-1)* vectors are built in this manner $X_{ij} = [0^T, 0^T, ..., X^T, ... - X^T, ..., 0^T]^T$ with dimensions $(l+1)M \times 1$. These vectors contain zero blocks, except in the cells *i,j* occupied by X^T and $-X^T$, respectively, $\forall j \neq i$.

2. We build the vector block $w = [w_1^T, w_2^T, ... w_M^T]^T$.

3. If X $\in$ ωi , that implies $w^T X_{ij} > 0, \forall j = 1,2,....M, j \neq i$.

The objective now is to design a linear classifier in the extended $(l+1)M \times 1$dimensional space, in such way that everyone of the $(M-1) \times N$ vectors lies on its positive side.

Chapter 4

PACKET CLASSIFIER

4.1 Generalized Classification Approach

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 a decision hyperplane that characterize the region of the space where each class is located:

$$g_j(x) = w_{0,j} + \sum_{i=1}^{M} w_{i,j} x_i, \qquad j = 1,2,..,C \quad , \tag{4.1}$$

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

The classifier's 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, using four of the classification methods described on Chapter 3, selected because of their simplicity and computational appeal: Pocket, Perceptron, LMS and SOE.

Each one of these methods provides a decision concerning classification and a geometrical interpretation of the job of discriminating amongst classes.

To begin with, features of each class are defined through their distribution probability functions, the algorithm foresees to work with features that follow distribution of these types: Normal, Rayleigh or Uniform; using training vectors that conform the characterization of each class. Once the training set for each class is defined, it is utilized for the construction of each classifier –regardless of the algorithm being developed.

Figures 4.1 and 4.2 illustrate examples of characterization of the 3 features that define each class, in a 2-Classes case *(C=2, M=3)*. Figure 4.1 represents characterization for class 1:

- Feature 1: Uniform Distribution, $U(1,2)$
- Feature 2: Normal Distribution, $N(2,0.64)$
- Feature 3: Uniform Distribution $U(1,5)$

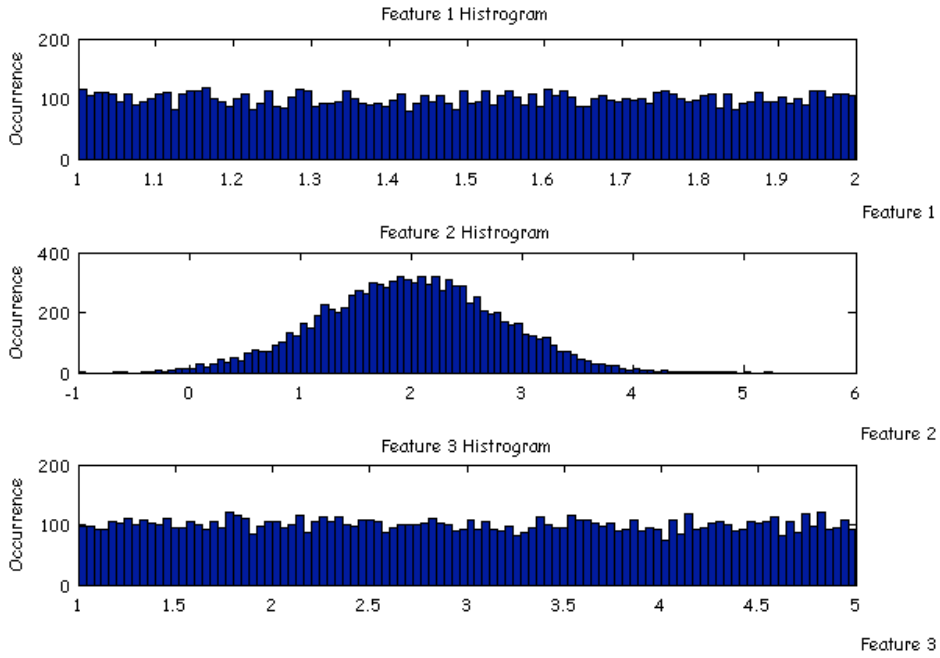


Figure 4.1 - Example of Characterization Stage (Class 1)

On the other hand, class 2 is characterized this way:

- Feature 1: Uniform Distribution, $U(2,3)$
- Feature 2: Rayleigh Distribution, Rayleigh (0.16)
- Feature 3: Uniform Distribution, $U(6,11)$

The following figure presents a graphic interpretation of this characterization:

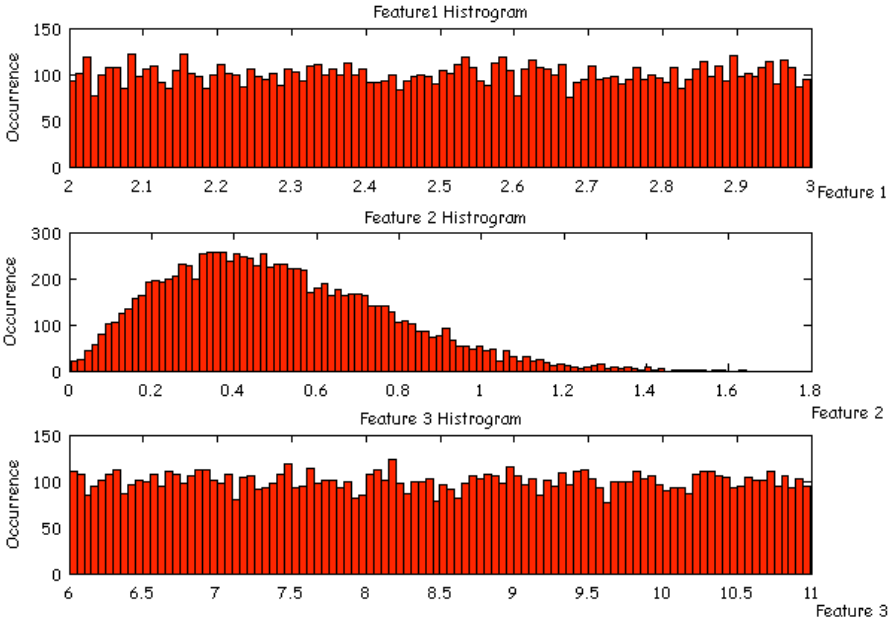


Figure 4.2 - Example of Characterization Stage (Class 2)

Once characterization of the classes through definition of each feature is accomplished, training vectors representing each class are built; where every vector has 3 components , with the values that correspond to each feature.

The following figure represents the corresponding features space:

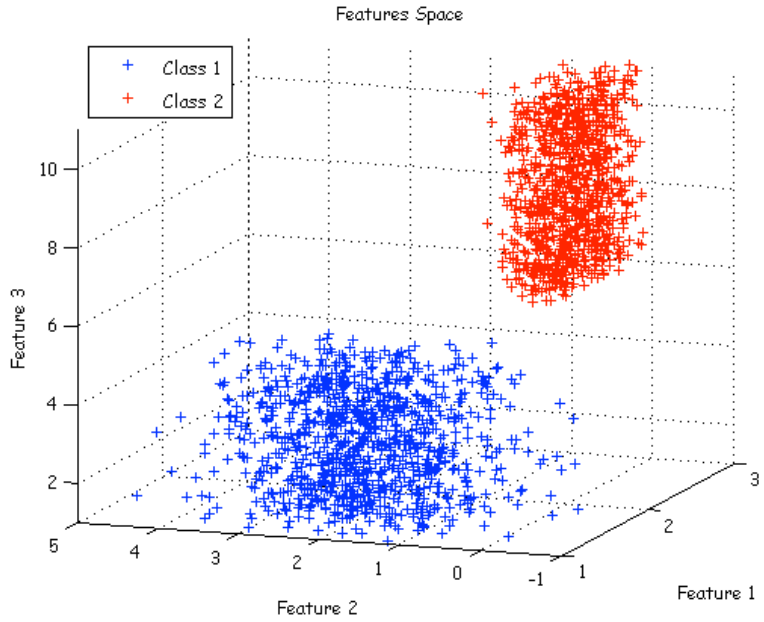


Figure 4.3 - Training Set Example ($C=2$, $M=3$)

For each of the implemented algorithms, a plane is built in which features space is divided in two regions; for the Perceptron algorithm's case, separation is illustrated as follows:

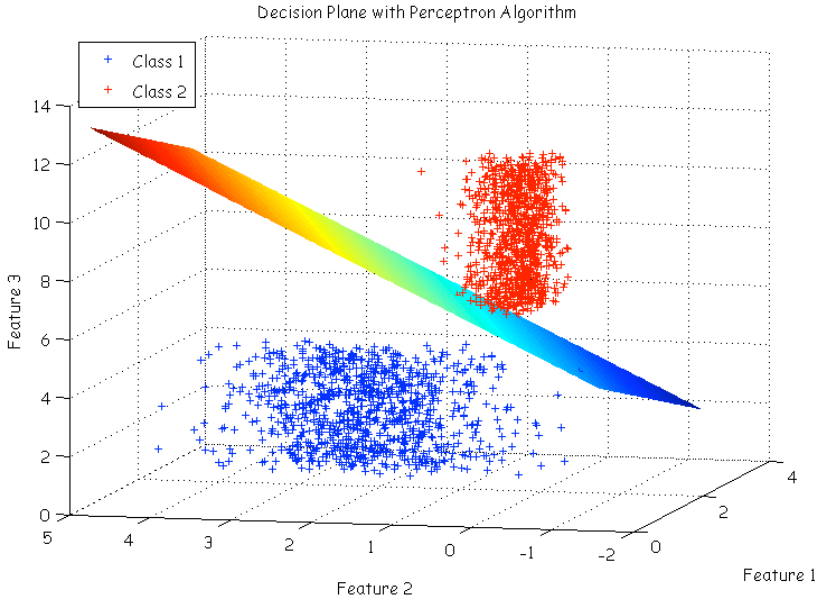


Figure 4.4 - Decision Plane according to Perceptron Algorithm

The solution for the Pocket Algorithm's case will look this:

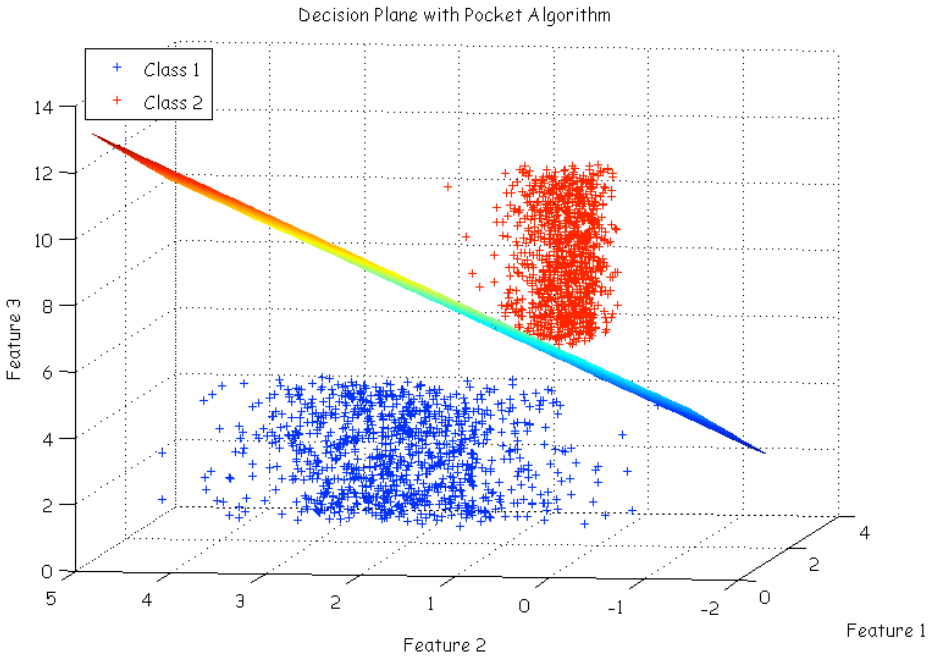


Figure 4.5 - Decision Plane according to Pocket Algorithm

The Figure 4.6 displays the LMS algorithm's decisions plane:

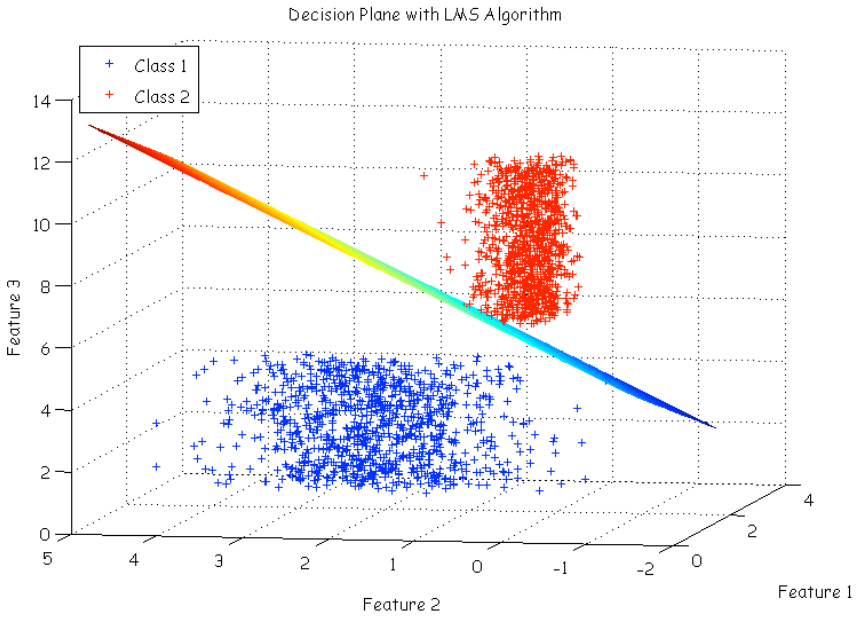


Figure 4.6 - Decision Plane according to LMS Algorithm

Finally, the SOE algorithm also displays a graphic interpretation of the calculated discriminant function:

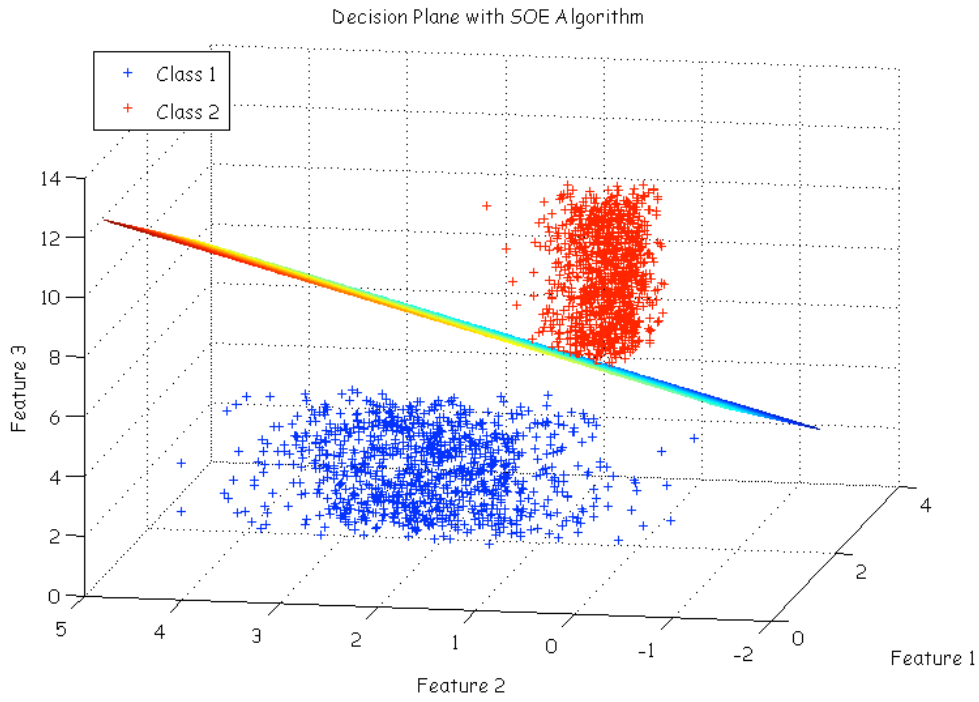


Figure 4.7 - Discriminant Function according to SOE Algorithm

4.2 Feature Selection

As explained on section 1.3, this work is oriented towards developing a module that can achieve automatic recognition of technologies and interference operating over the ISM bands, specifically in the 2.4 GHz band. This classification block is one of the cornerstones of the AIR-AWARE Project. The project's objective is to create a black box — the AIR-AWARE module — capable of classifying technologies, as well as different types of interference in play [8].

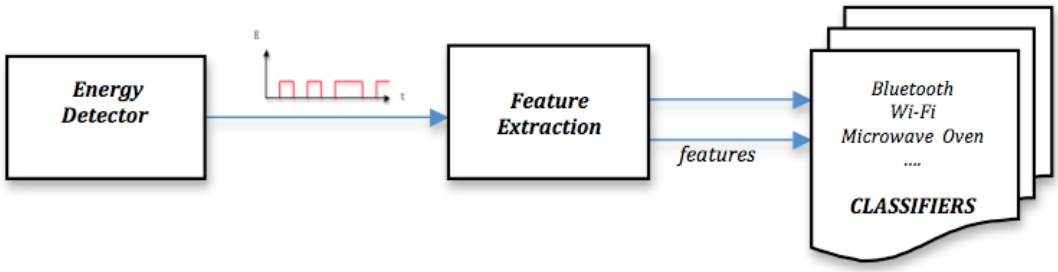


Figure 4.8 - The AIR-AWARE module

As figure 4.8 displays, a beginning block consists of a device capable of spectrum sensing, with a good time resolution (in the μsec order). This device is not in capacity to demodulate or decode, but it is able to provide reliable information about presence or absence of energy over time. The design of this block is outside the realm of this project, however the following figure show two types of implementation of an Energy Detector: the first a Conventional energy detector [9], consisting of a low pass filter to reject out of band noise and adjacent signals, Nyquist sampling A/D converter, square-law device and integrator (Figure 4.9 (a)).

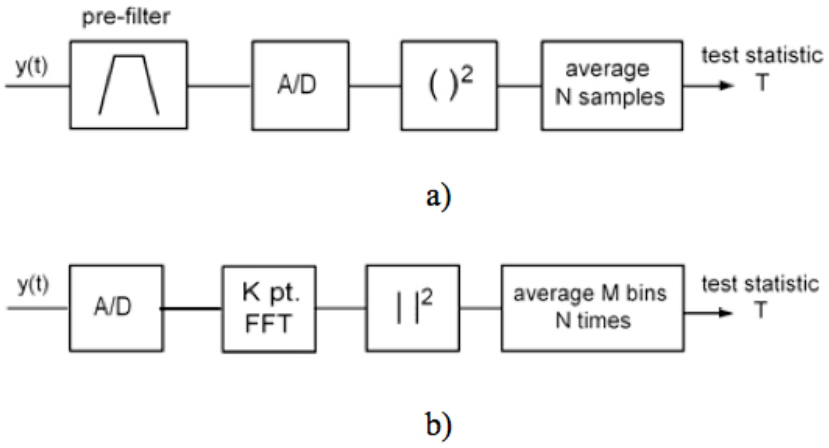


Figure 4.9 - Energy Detector

In contrast, figure b represents an alternative approach which could be devised by using a periodogram to estimate the spectrum via squared magnitude of the FFT.

The next block of the AIR-AWARE module is feature extraction stage. To achieve

recognition of technologies, a set of features should be defined; this means, each pattern to classify must be translated to a vector that contains the appropriate characterizing features. Feature selection depends upon the fundamental characteristics of the classes to classify; a good candidate will be a characteristic with great discriminatory power between classes.

This study is focused on 802.11 (Wi-Fi) and 802.15.1 (Bluetooth) network recognition. Previous work, as [1], 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.

The selection of features phase was took over in [10] through extensive analysis of MAC sublayer communication procedures based on real Wi-Fi traffic, where the main goal was to identify MAC sublayer [1,2] specific features for each of the above technologies and, through these, achieve differentiation.

The automatic recognition approach proposed on this work is based on the utilization of two features: the SIFS, with high discriminatory richness [10] and the Maximum Packet Duration between two silence Gaps.

The first feature is the time interval between PPDUs, defined in Section 2.3 as Short Inter Frame Space. Of all existing IFS types, SIFS has a nominal value of $10\mu s$ 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.

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 by using the simplest hardware: an energy detector.

Once features are selected: Duration of Silence Gaps (*feature 1)* and Maximum Packet Duration between two Silence Gaps (*feature 2*); the module Feature Extraction must be able to extract each of these parameters in any sequence of packets at its input. From here, every packet sequence is translated into a set of point on a bidimensional plane with

coordinates [*x,y*], where the x axis represents the Duration of Silence Gaps and the y axis represents the Maximum Packet Duration between two Silence Gaps.

4.3 Experimentation

4.3.1 Training Set Construction

As detailed in Chapter 3, before implementation of any linear classification algorithm is possible, a strong set of characteristic vectors for each class is necessary: a training set. Before anything else, to achieve Wi-Fi vs. Bluetooth recognition:

- ✓ Wi-Fi real traffic was utilized.
- ✓ Bluetooth traffic was simulated considering [12] the case of a Piconet with two devices in connection state (one master and one slave). 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. 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. According to [12] the jitter was modeled by a Gaussian distribution with zero mean and standard deviation s = 10/3μs.

  To enhance variability in the generation of this packet sequence, two different scenarios were considered on the first one, 100% of transmitted packets occupy just one Time Slot; whereas the second scenario represents higher variability with 80% of the packets occupying 1 Time Slot, while 15% represent 3 Time Slot duration and 5% occupy % Time Slots. In every scenario, 70 % of the data packets have a duration that is fixed by the protocol [11].The duration of the remaining 30% is uniformly distributed between minimum and maximum values [12].

	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		

Table 4.1 - Bluetooth Standard Specifications

For the construction of the training set that characterizes each class, six 1000-packet captures for Wi-Fi and two 6000-packet MATLAB simulated Bluetooth captures –single slot or multi slot for each scenario- were utilized. Each capture was translated into a set of points (2093 points) after the feature extraction stage. This sets were then located on a bidimensional space, reflecting characteristic regions of each class on the plane. The following figures illustrate the respective training sets, for the Bluetooth multi slot and single slot communications cases.

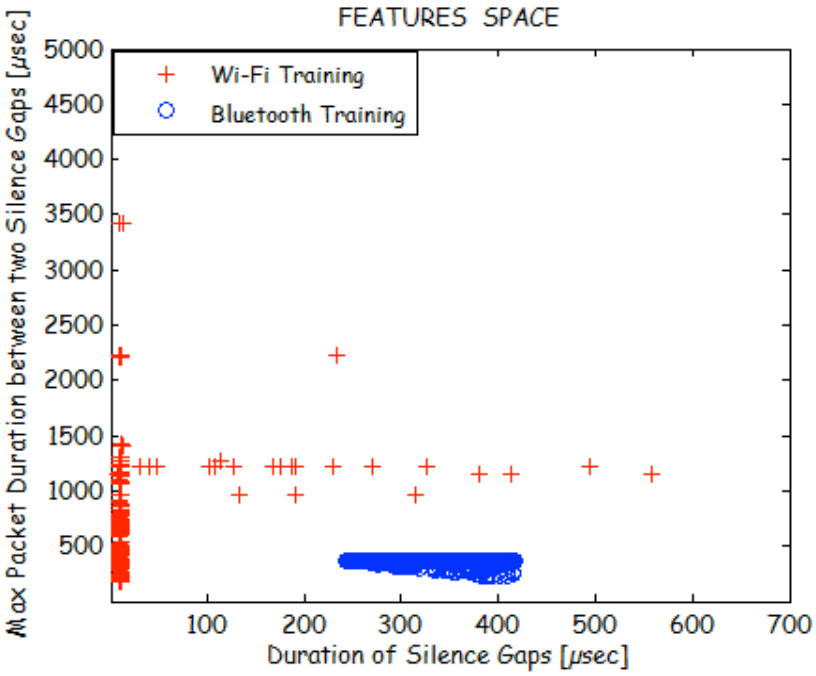


Figure 4.10 - Features Plane with Single-Slot Communication at Bluetooth Class

Because in Figure 4.10 represents the Bluetooth single-slot case, the corresponding training set for the Bluetooth class displays lower variability and therefore lower dispersion amongst the points that constitute that class; this in turn favors separability of the classes to classify.

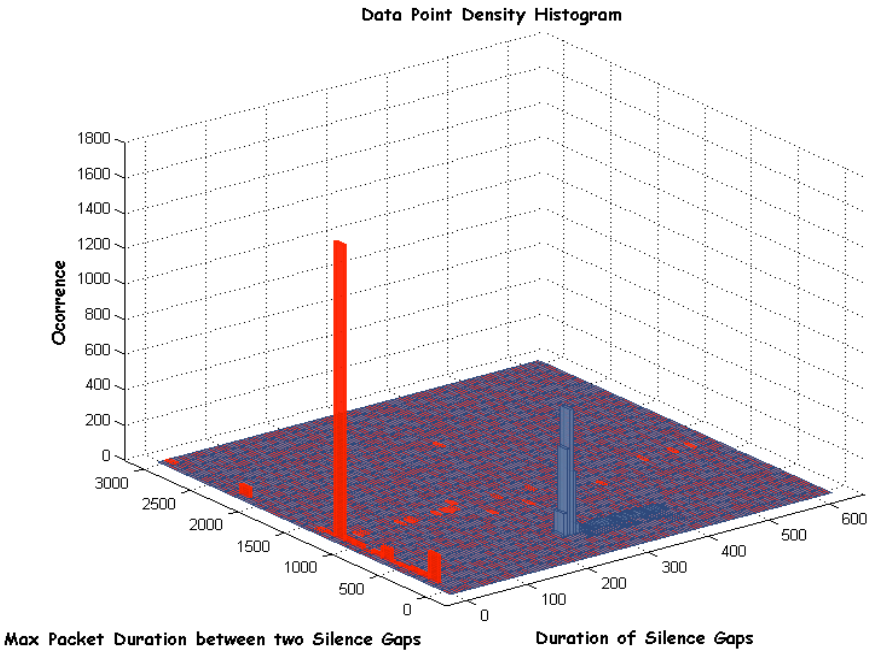


Figure 4.11 - Data Point Density Histogram with Single-Slot Communication at Bluetooth Class

Through Figure 4.11, we are able to verify how the high density of points corresponding to the Wi-Fi class have a Silence Gaps duration around 10 μsec; whereas for the Bluetooth class, in the single slot scenario, duration of silence gaps fluctuates between 200 and 400 μsec, while maximum packet duration remains always lower than 500 μsec.

However, the following figure, representing the multi-slot case, presents a less restricted training set behavior. In fact the presence of a few Wi-Fi points invading the Bluetooth "zone" becomes noticeable. Nonetheless, capture file revision indicated that these corresponded to non-SIFS, i.e. erroneously estimated SIFS. In any case, these points were less than 1% of total.

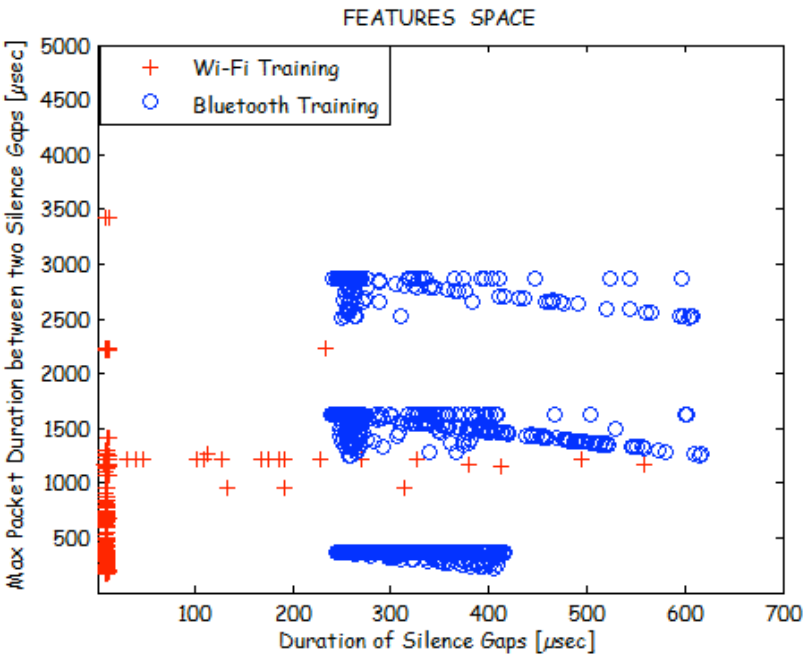


Figure 4.12 - Features Plane with Multi-Slot Communication at Bluetooth Class

The next figure presents a data point density histogram, useful to comprehend further the distribution of these points and their proportion in respect to the training set.

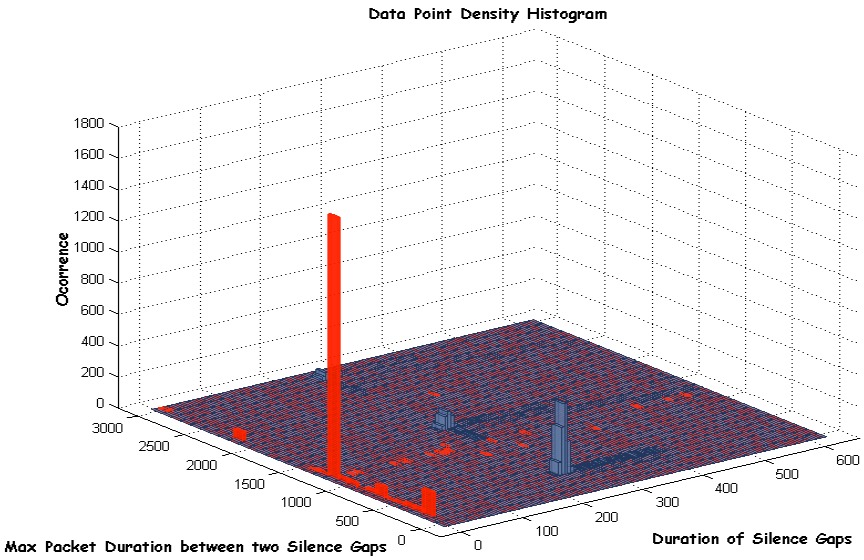


Figure 4.13 - Data Point Density Histogram with Multi-Slot Communication at Bluetooth Class.

Afterwards, when training sets were defined, each of the chosen classification algorithms was implemented: Pocket, Perceptron, LMS and SOE. Graphic interpretation of each the algorithms is presented as follows, differentiating between the single slot and multi slot cases.

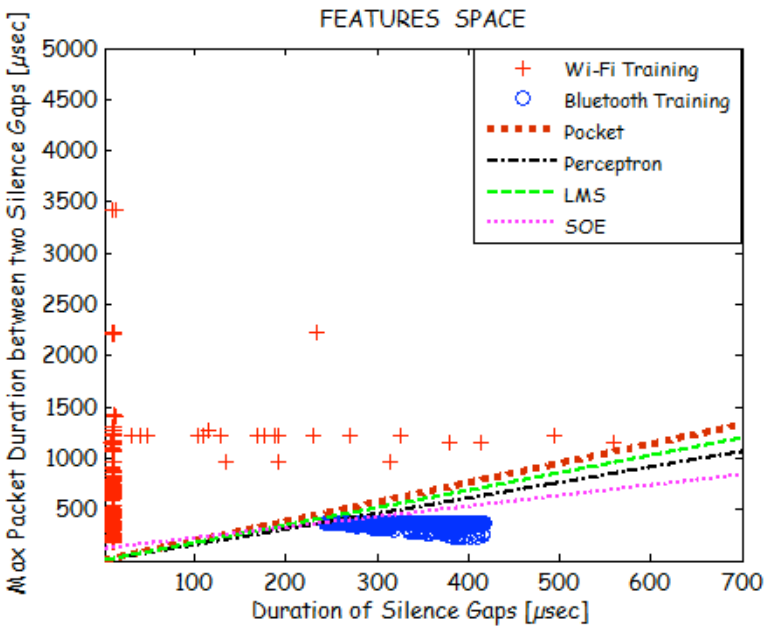


Figure 4.14 Automatic classification of Wi-Fi vs. Bluetooth Single-Slot

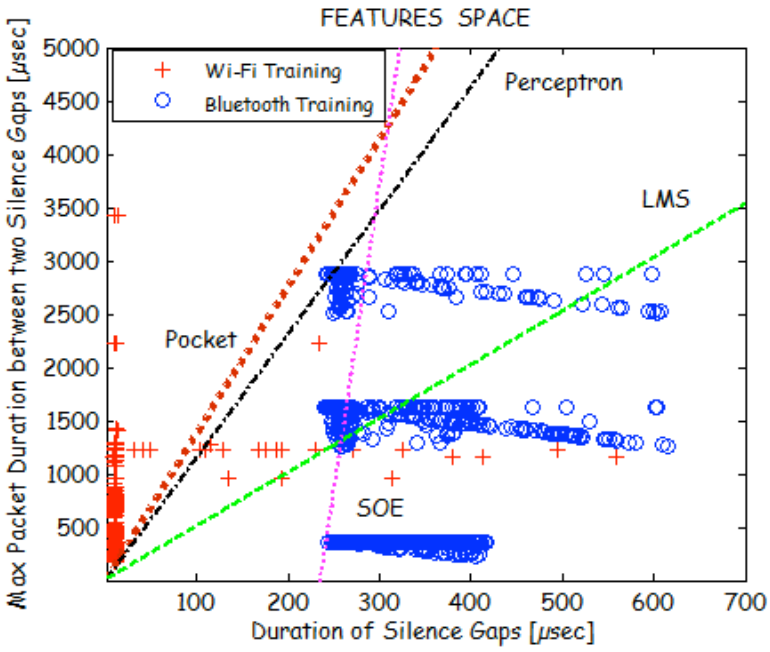


Figure 4.15 Automatic classification of Wi-Fi vs. Bluetooth Multi-Slot.

It becomes evident –when confronting the single and multi slot scenarios- that due to the separability of the classes present on the first scenario all methods reach convergence and therefore all training vectors- each corresponding to a class- are classified correctly. On the contrary, the second scenario presents a much more complex set to classify –due to the scarce separability of the classes- in this case the utilized classification method classifies erroneously a percentage of the points. The following table presents the error percentages of each algorithm:

	% Wi-Fi vectors erroneous	***% Bluetooth vectors erroneous***
Pocket	1% [22/2192]	0% [0/2192]
Perceptron	1% [22/2192]	0.32% [7/2192]
LMS	0.5% [11/2192]	31.75% [696/2192]
SOE	0.5% [11/2192]	26.27% [576/2192]

Table 4.2 Percentage of Error over the Training Set for each algorithm -Wi-Fi and Bluetooth Multi-slot Communication Class-.

To add fortitude to the classification and analyze the percentage of classification errors in regards to the number of training vectors, a second strategy with a higher number of training vectors was utilized. For the Wi-Fi case, seventeen (16) 1000-packet captures were utilized. For the Bluetooth case, simulated MATLAB captures consisted in two 20000-packet sequences corresponding to either single slot or multi slot cases. In this case, features space is boarded by a higher quantity of points (in order to 9000). Following, an illustration of the respective features space, for each Bluetooth scenario considered:

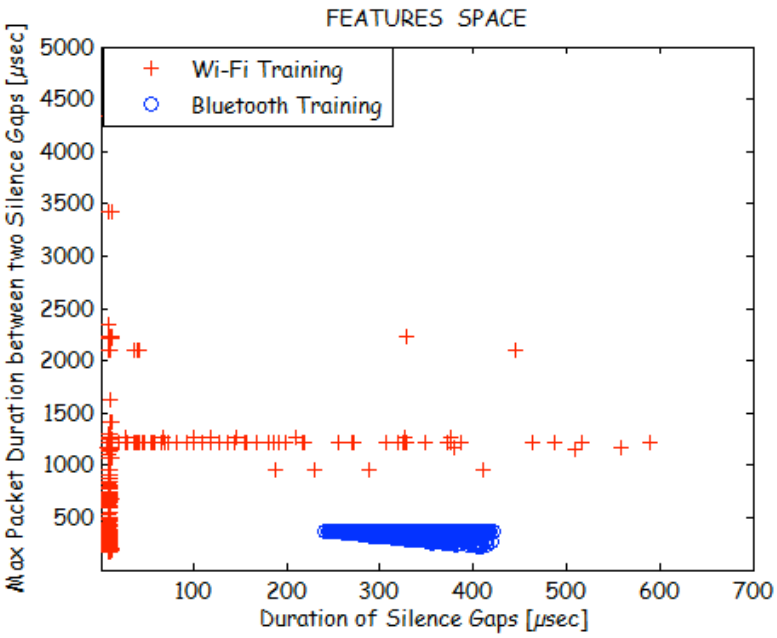


Figure 4.16 - Features Plane with Single-Slot Communication at Bluetooth Class (*double dimension training*)

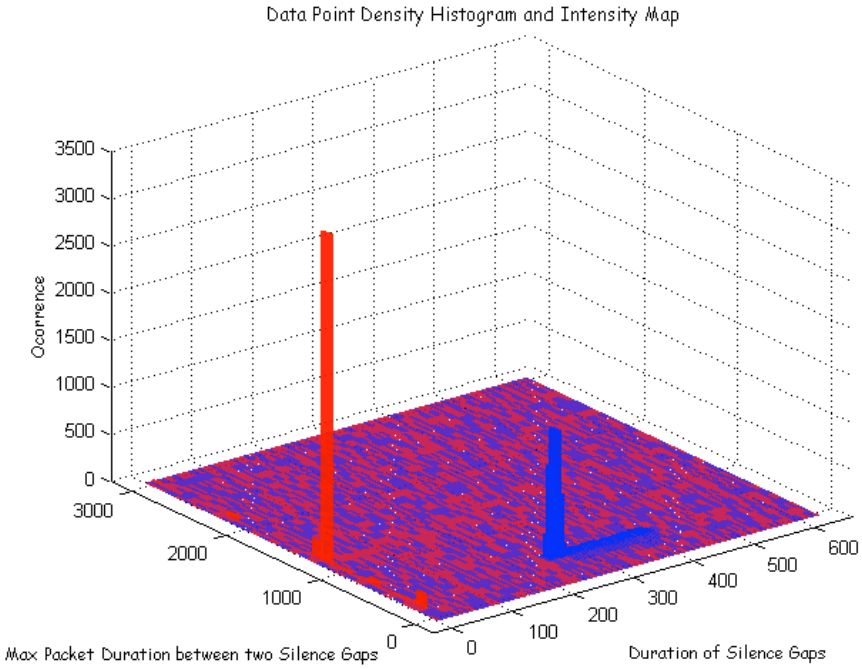


Figure 4.17 - Data Point Density Histogram with Single-Slot Communication at Bluetooth Class. (*double dimension training*)

The resulting training set for the Multi-Slot case:

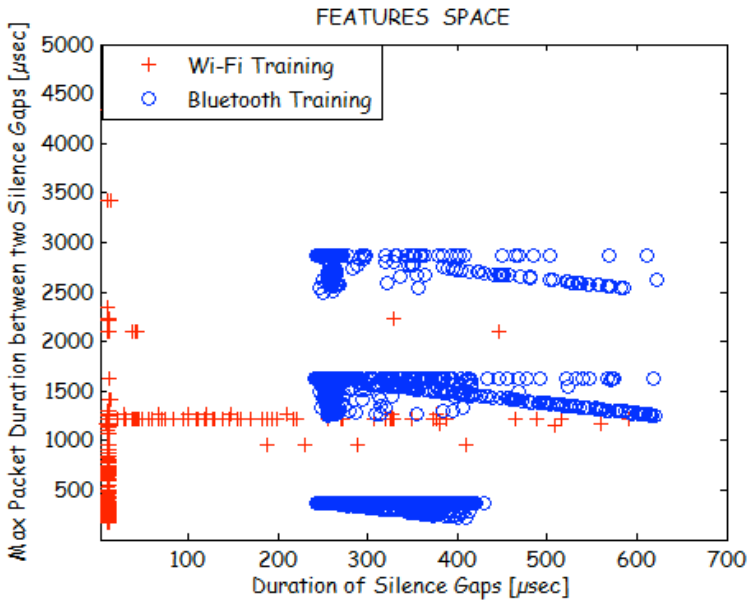


Figure 4.18 - Features Plane with Multi-Slot Communication at Bluetooth Class (*double dimension training*)

The corresponding data point density is displayed as follows:

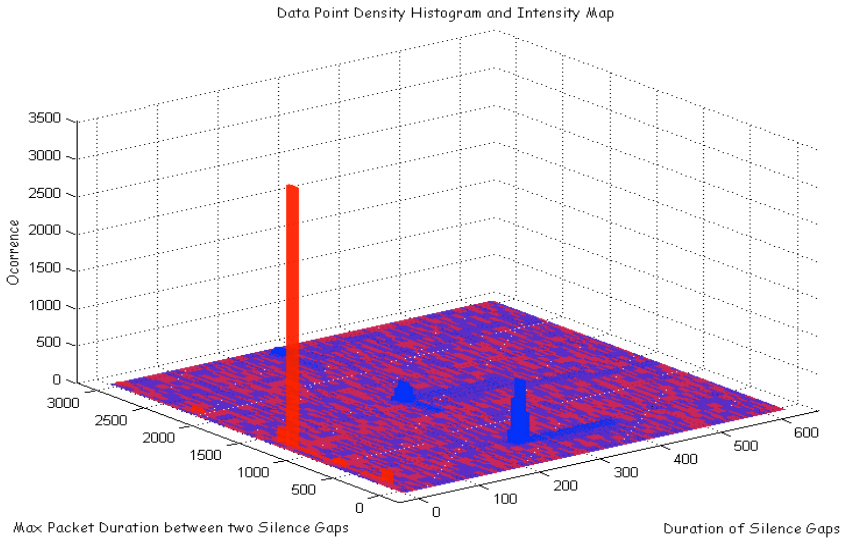


Figure 4.19 - Data Point Density Histogram with Multi-Slot Communication at Bluetooth Class (*double dimension training*)

The following figures illustrates the implementations of the classification algorithms, in this considering a training set of broader dimensions.

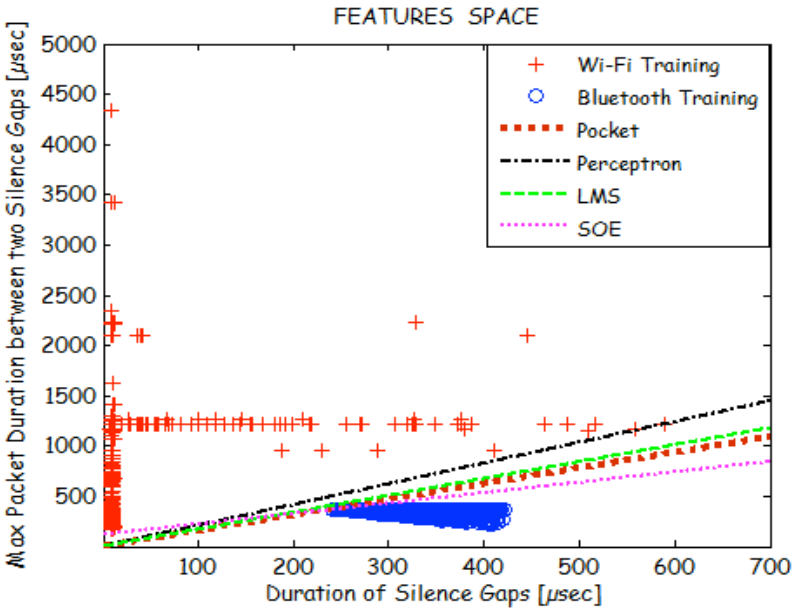


Figure 4.20 - Automatic classification of Wi-Fi vs. Bluetooth Single-Slot

(*double dimension training*)

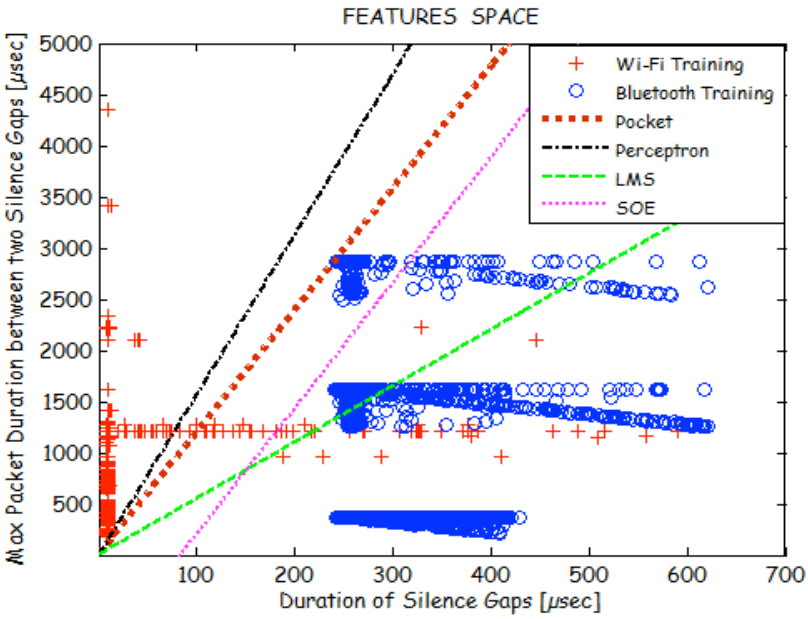


Figure 4.21 - Automatic classification of Wi-Fi vs. Bluetooth Multi-Slot

(*double dimension training*)

Again, separability of the classes favors convergence of the classification algorithms on the first scenario, while hurdling complete separation of the training vectors in the second scenario, resulting in the following percentages of error per each algorithm:

	% Wi-Fi vectors erroneous	***% Bluetooth vectors erroneous***
Pocket	1.23% [53/4294]	0.02% [1/4294]
Perceptron	1.56% [67/4294]	0% [0/4294]
LMS	0.63% [27/4294]	31.8% [1366/4294]
SOE	0.79% [34/4294]	8.17% [351/4294]

Table 4.3 - Percentage of Error for each algorithm over the Training Set Wi-Fi and Bluetooth Multi-slot Communication Class. (Double dimension training)

4.3.2 Classification Results

In order to test each of them, the implemented 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.

It bears weight, as explained before, that each sequence to classify is first processed by the feature extraction block, and therefore translated into sets of points onto the features space; where the utilized sequences correspond to 352 points for the Wi-Fi case (1000 packets), and 456 points for the Bluetooth case. Percentages on the subsequent tables are based on this number of points, where for each case (being the input either Wi-Fi or Bluetooth), and according to each algorithm, the percentage of points distinguished as belonging to either of the classes is reflected.

	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]

Table 4.4 - Classification results with single-slot Communications at Bluetooth class.

The results of automatic classification considering the second Bluetooth scenario Multi –Slot are display on the following Table.

	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 4.5 - Classification results with multi-slot Communications at Bluetooth class

It became interesting to study how automatic recognition will operate when the input was a mixed flow (multi-network environment). 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 mixes were generated:

1. pre-dominant Wi-Fi (1000 Wi-Fi packets vs. 200 Bluetooth packets);
2. balanced (1000 Wi-Fi packets vs. 1000 Bluetooth packets);
3. Bluetooth pre-dominant (1000 Wi-Fi vs. 2000 Bluetooth packets).

The same considerations where applied to the single and multi slot Bluetooth scenarios, obtaining the results shown as follows:

	Input Network	*Classification into Wi-Fi*	*Classification into single-slot Bluetooth*
Pocket	*Bluetooth pre-dominant*	38.32% [292/762]	61.68% [470/762]
Pocket	*Wi-Fi pre-dominant*	86.45% [351/406]	13.55% [55/406]
Pocket	*Balanced*	58.21% [303/520]	41.79% [217/520]
Perceptron	*Bluetooth pre-dominant*	38.45% [293/762]	61.55% [469/762]
Perceptron	*Wi-Fi pre-dominant*	86.7% [352/406]	13.3% [54/406]
Perceptron	*Balanced*	57.83% [301/520]	42.17% [219/520]
LMS	*Bluetooth pre-dominant*	38.32% [292/762]	61.68% [470/762]
LMS	*Wi-Fi pre-dominant*	86.45% [351/406]	13.55% [55/406]
LMS	*Balanced*	58.21% [303/520]	41.79% [217/520]
SOE	*Bluetooth pre-dominant*	38.58% [294/762]	61.42% [468/762]
SOE	*Wi-Fi pre-dominant*	86.70% [352/406]	13.3% [54/406]
SOE	*Balanced*	58.21% [303/520]	41.79% [217/520]

Table 4.6 - Classification results with single-slot Communications at Bluetooth class. (Mixed Input)

	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]

Table 4.7 - Classification results with multi-slot Communications at Bluetooth class. (Mixed Input)

Also, results from classification are reflected, related to a training set of higher dimension (approximately double in size). In order to test the classification algorithms a 1000-packet Wi-Fi capture and 2 relative sequences of 1000 Bluetooth packets were utilized – in both single-slot and multi-slot cases-.

	Classifier Input Network	*Classification into Wi-Fi*	*Classification into single-slot Bluetooth*
Pocket	*Bluetooth*	0% [0/440]	100% [440/440]
Pocket	*Wi-Fi*	100% [355/355]	0% [0/355]
Perceptron	*Bluetooth*	0% [0/440]	100% [440/440]
Perceptron	*Wi-Fi*	100% [355/355]	0% [0/355]
LMS	*Bluetooth*	0% [0/440]	100% [440/440]
LMS	*Wi-Fi*	100% [355/355]	0% [0/355]
SOE	*Bluetooth*	0% [0/440]	100% [440/440]
SOE	*Wi-Fi*	100% [355/355]	0% [0/355]

Table 4.8 - Classification results with single-slot Communications at Bluetooth class. (Double dimension training)

The results of automatic classification considering the Bluetooth scenario multi slot are:

	Classifier Input Network	*Classification into Wi-Fi*	*Classification into multi-slot Bluetooth*
Pocket	*Bluetooth*	0% [0/447]	100% [447/447]
Pocket	*Wi-Fi*	98.59% [350/355]	1.41% [5/355]
Perceptron	*Bluetooth*	0% [0/447]	100% [447/447]
Perceptron	*Wi-Fi*	98.31% [349/355]	1.69% [6/355]
LMS	*Bluetooth*	33.3% [149/447]	66.67% [298/447]
LMS	*Wi-Fi*	99.44% [353/355]	0.56% [2/355]
SOE	*Bluetooth*	8.73% [39/447]	91.28% [408/447]
SOE	*Wi-Fi*	98.87% [351/355]	1.13% [4/355]

Table 4.9 - Classification results with multi -slot Communications at Bluetooth class (*Double dimension training*)

In much the same way, with the goal of analyzing the performance of the classifier when training set dimension varies, a simulation of mixed packet flow was generated (simulated multi-network environment); the considerations of this generation match those of the previous case.

Results of the single-slot and multi-slot cases are displayed in the following tables.

	Input Network	*Classification into Wi-Fi*	*Classification into single-slot Bluetooth*
Pocket	*Bluetooth pre-dominant*	36.72% [282/768]	63.28% [486/768]
Pocket	*Wi-Fi pre-dominant*	86.52% [353/408]	13.48% [55/408]
Pocket	*Balanced*	57.14% [324/567]	42.86% [243/567]
Perceptron	*Bluetooth pre-dominant*	36.07% [277/768]	63.93% [491/768]
Perceptron	*Wi-Fi pre-dominant*	86.03% [351/408]	13.97% [57/408]
Perceptron	*Balanced*	56.97% [323/567]	43.03% [244/567]
LMS	*Bluetooth pre-dominant*	36.46% [280/768]	63.54% [488/768]
LMS	*Wi-Fi pre-dominant*	86.27% [352/408]	13.73% [56/408]
LMS	*Balanced*	57.14% [324/567]	42.88% [243/567]
SOE	*Bluetooth pre-dominant*	36.71% [282/768]	63.28% [486/768]
SOE	*Wi-Fi pre-dominant*	86.52% [353/408]	13.48% [55/408]
SOE	*Balanced*	57.14% [324/567]	42.88% [243/567]

Table 4.10 - Classification results with single-slot Communications at Bluetooth class. (Mixed) (*Double dimension training*)

	Input Network	*Classification into Wi-Fi*	*Classification into multi-slot Bluetooth*
Pocket	*Bluetooth pre-dominant*	17.35% [136/784]	82.65% [648/784]
Pocket	*Wi-Fi pre-dominant*	82.25% [329/400]	17.75% [71/400]
Pocket	*Balanced*	42.09% [213/506]	57.91% [293/506]
Perceptron	*Bluetooth pre-dominant*	16.45% [129/784]	83.55% [655/784]
Perceptron	*Wi-Fi pre-dominant*	82.25% [329/400]	17.75% [71/400]
Perceptron	*Balanced*	41.3% [209/506]	58.7% [297/506]
LMS	*Bluetooth pre-dominant*	40.05% [314/784]	59.95% [470/784]
LMS	*Wi-Fi pre-dominant*	86.25% [346/400]	13.75% [55/400]
LMS	*Balanced*	60.87% [308/506]	39.13% [198/506]
SOE	*Bluetooth pre-dominant*	22.70% [178/784]	77.30% [606/784]
SOE	*Wi-Fi pre-dominant*	83.75% [335/400]	16.25% [65/400]
SOE	*Balanced*	47.23% [239/506]	52.77% [267/506]

Table 4.11 Classification results with multi-slot Communications at Bluetooth class. (Mixed) (*Double dimension training*)

Conclusions and Future Work

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. We were able to conclude the following from the results obtained during the experimentation stage:

1. For the Wi-Fi vs. single-slot Bluetooth case 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, 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. For the mixed flow experimentation, results point out 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 from the single-slot and multi-slot cases. We were able to determine that only the Pocket and Perceptron algorithms are capable of performing a reliable classification. Results point out to plausible detection when both technologies are present simultaneously. To improve accuracy of decision, post processing will be required .

Future work could 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) 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 $192\mu s$ [16] 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] Gandetto M. and Regazzoni C., "Spectrum Sensing: A Dsitributed Approach for Cognitive Terminals," IEEE Journal on selected areas in communications, Vol.25 (3), 2007.

[2] http://it.wikipedia.org/wiki/Wireless_Local_Area_Network.

[3] 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.

[4] http://en.wikipedia.org/wiki/ISM_band.

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

[8] Di Benedetto M.G, Boldrini S., Martin Martin C.J, Roldan Diaz. J. . 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.

[9] Cabric D., Tkachenko A. and Brodersen R.. "Experimental Study of Spectrum Sensing based on Energy Detection and Network Cooperation", in TAPAS 2006.

[10] Roldán Jesus. " Cognitive Networking: Network Sensing with Application to IEEE 802.11 Communication Systems". M.S degree in Telecommunication Engineering, Sapienza Università di Roma, 12 April 2010.

[11] 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.

[12] Boldrini, Stefano. "Recognition of Bluetooth Signals based on Feature Detection". M.S degree in Telecommunication Engineering, Sapienza Università di Roma, 12 April 2010.

[13] http://es.wikipedia.org/wiki/Radio_cognitiva/

[14] http://en.wikipedia.org/wiki/Electromagnetic_interference_at_2.4_GHz

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

[16] 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.

Appendix

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.

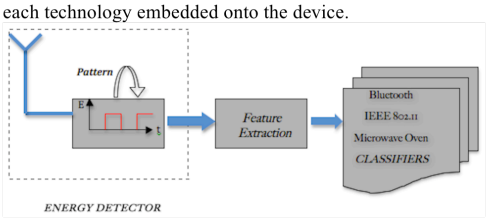


Figure 1 - The AIR-AWARE module

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 $10\mu s$ 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).

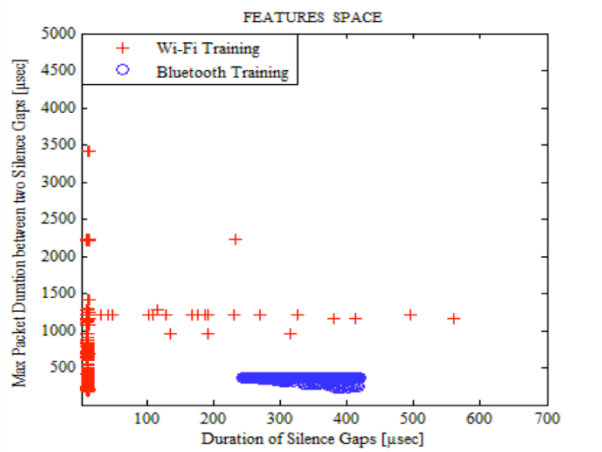


Figure 2 - Feature Plane for Wi-Fi and Bluetooth single-slot

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.

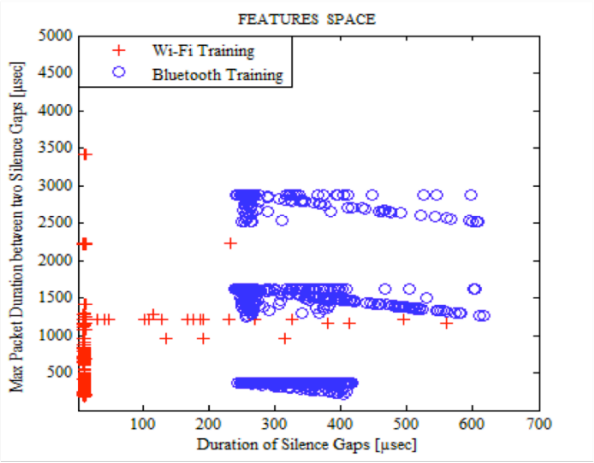


Figure 3 - Feature Plane for Wi-Fi and Bluetooth multi-slot

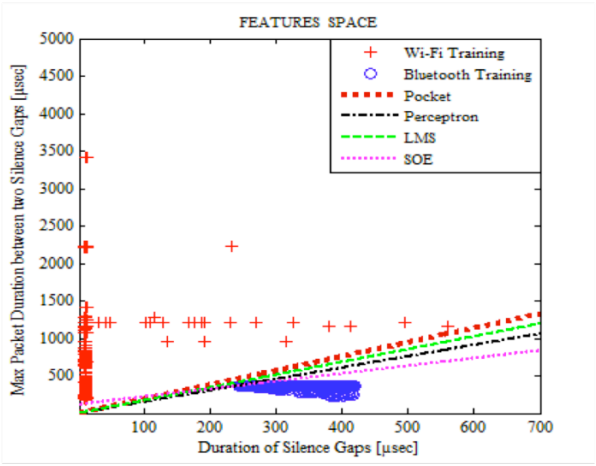


Figure 4 - Automatic classification of Wi-Fi vs. Bluetooth single-slot

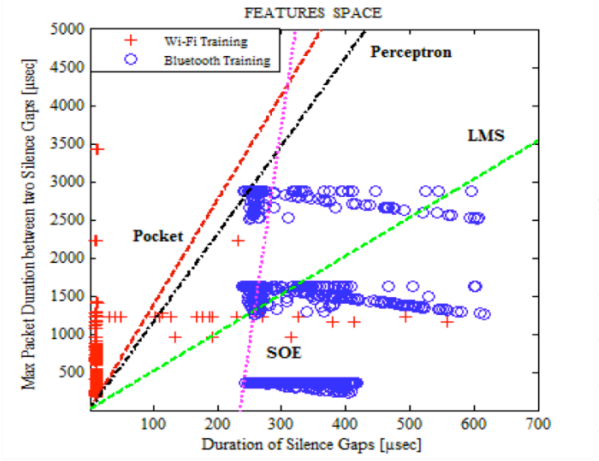


Figure 5 - Automatic classification of Wi-Fi vs. Bluetooth multi-slot

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.

Generalized Classifier Code:

1. Training2:

```
%Generalized Classifier Script


%Generating Features second Probability Density Function

C=input('\n Number of Classes to be classifier:') ; % Number of Classes

M=input('\n Number of Features:'); % Number of Features

N=input('\n Choose the number of training for only Class:') ; % Training
Number

fprintf('\n -Gaussian Distribution = 1 ')

fprintf('\n -Rayleigh Distribution = 2 ')

fprintf('\n -Uniform Distribution = 3 \n')

fprintf('\n Choose second a number the distributions for each feature \n')

Distributions=zeros(M,C);

for c=1:C

    fprintf('\n * Choose the feature distribution for the Class %d \n ',c)

     for f=1:M

         fprintf('\n  For the feature: %d \n',f)

         Distributions(f,c)=input('\n Choose the distribution:');

     end
end


% Feature's Characterization

Distributionx=Distributions;

Vector_parameters=zeros(5,C*M);

for d=1:C

    fprintf('\n * Insert the parameters for Class %d\n ',d)

  Distributions_1= Distributions(1:M,d);


  for g=1:M

      fprintf('\n Characterization of Feature %g\n',g)

      switch (Distributions_1(g))

          case 1

          Vector_parameters(1,g+M*(d-1))=input('\n Insert sigma:');
```

```
          Vector_parameters(2,g+M*(d-1))=input('\n Insert mean:');

          case 2

              Vector_parameters(3,g+M*(d-1))=input('\n Insert sigma:');

          case 3

              Vector_parameters(4,g+M*(d-1))=input ('\n Insert the start
interval:');

              Vector_parameters(5,g+M*(d-1))=input('\n Insert the end
interval:');
      end

  end

end

Parameters=Vector_parameters;

Feature_matrix=zeros(N,C*M);


%Calculating the Feauture's Histogram

for t=1:C

    Distributions_1= Distributions(1:M,t);


    for y=1:M

        switch (Distributions_1(y))

          case 1

              Sigma=Parameters(1,y+M*(t-1));

              Mean=Parameters(2,y+M*(t-1));

              Feature_matrix(1:N,y+M*(t-1))= randn(1,N)*Sigma +Mean;


            case 2

                Uniform_Distribution=rand(1,N);

                Sigma=Parameters(3,y+M*(t-1));

                    for i=1:N

                        Feature_matrix(i,y+M*(t-1))= Sigma*sqrt(-2*log(1-
Uniform_Distribution(i)));

                    end
            case 3

                a=Parameters(4,y+M*(t-1));

                b=Parameters(5,y+M*(t-1));


                        Feature_matrix(1:N,y+M*(t-1))= a + (b-a).*
```

```
rand(N,1);
        end

    end

end

X1=Feature_matrix; % Training Matrix


% Plotting Histograms

    for l=1:C

        figure(l)


         for z=1:M

             subplot(M,1,z),hist(Feature_matrix(1:N,z+M*(l-1)),100)

             title(['Characterization of Class',int2str(l) '.
Feature',int2str(z) ' Histrogram'])

             xlabel(['Feature ',int2str(z)]);

             ylabel('Occurrence');


         end


    hold on

    end


    % Folder Name's Construction

    FolderName=cbuild(C,M);

    mkdir( FolderName);


    % Plotting the M-dimensional Space



    ColorSet=[0 0 1 ; 1 0 0 ; 0 1 0 ; 0 1 1 ; 1 0 1 ;0 0 0;1 1 0];

    controlC=C-1;

          switch(M)

              case 1
```

```
                        figure(C+1)
                       for f=1:M:M*C
p1=plot(X1(1:N,f),zeros(1,N),'+','Color',ColorSet(1+((f-1)/M),:));
                            hold on
                       end
                        hold on
                       title('Features Space')

                        switch(controlC)
                            case 1
                            legend('Class 1','Class 2',2);
                            case 2
                            legend('Class 1','Class 2','Class 3',3);
                            case 3
                            legend('Class 1','Class 2','Class 3','Class
4',4);
                            case 4
                            legend('Class 1','Class 2','Class 3','Class
4','Class 5',5);
                            case 5
                            legend('Class 1','Class 2','Class 3','Class
4','Class 5','Class 6',6);
                            case 6
                            legend('Class 1','Class 2','Class 3','Class
4','Class 5','Class 6','Class 7',7);
                        end

                        xlabel('Feature')
                        ylim([-0.05 0.05])
                        grid on
                        cd (FolderName);
                        hgsave('1Feature')
                        cd ..

              case 2
```

```
                            figure(C+1)

                            for f=1:M:M*C

p2=plot(X1(1:N,f),X1(1:N,f+1),'+','Color',ColorSet(1+((f-1)/M),:));

                                hold on

                            end

                            hold on

                            title('Features Space')

                        switch(controlC)

                            case 1

                            legend('Class 1','Class 2',2);

                            case 2

                            legend('Class 1','Class 2','Class 3',3);

                            case 3

                            legend('Class 1','Class 2','Class 3','Class
4',4);

                            case 4

                            legend('Class 1','Class 2','Class 3','Class
4','Class 5',5);

                            case 5

                            legend('Class 1','Class 2','Class 3','Class
4','Class 5','Class 6',6);

                            case 6

                            legend('Class 1','Class 2','Class 3','Class
4','Class 5','Class 6','Class 7',7);

                        end

                        xlabel('Feature 1')

                        ylabel('Feature 2')

                        grid on

                        cd(FolderName)

                        hgsave('2Features')

                        cd ..

            case 3

                                figure(C+1)
```

```
                            for f=1:M:M*C

p3=plot3(X1(1:N,f),X1(1:N,f+1),X1(1:N,f+2),'+','Color',ColorSet(1+((f-
1)/M),:));
                                hold on
                            end
                            hold on
                            title('Features Space')
                        switch(controlC)
                            case 1
                            legend('Class 1','Class 2',2);
                            case 2
                            legend('Class 1','Class 2','Class 3',3);
                            case 3
                            legend('Class 1','Class 2','Class 3','Class
4',4);
                            case 4
                            legend('Class 1','Class 2','Class 3','Class
4','Class 5',5);
                            case 5
                            legend('Class 1','Class 2','Class 3','Class
4','Class 5','Class 6',6);
                            case 6
                            legend('Class 1','Class 2','Class 3','Class
4','Class 5','Class 6','Class 7',7);
                        end

                        xlabel('Feature 1')
                        ylabel('Feature 2')
                        zlabel('Feature 3')
                        grid on
                        cd(FolderName)
                        hgsave('3Features')
                        cd ..
          end


   % Generation matrix of Training (M,N*C) dimension
```

```
  D=N*C;% Total Number of Training

  Xp=zeros(M,N); %Training Matrix

  X_convertion= (X1)';

  Xq=zeros(M,N*C);

  X=zeros(1:M,N*C);



  for k=1:C

       Xp(1:M,1:N)= X_convertion(1+M*(k-1):(k*M),1:N);

       Xq=[Xq Xp];

       Xp=zeros(M,N);

  end

X=Xq(1:M,(N*C)+1:2*(N*C));

% Axis Construction

max_values=max(X,[],2);

min_values=min(X,[],2);


% Vector of Desired Response Generation

     if (C==2)

      y_ort=[ones(1,N) (ones(1,N)*(-1))];

     else

      y_ort=[ones(1,N) zeros(1,N*(C-1))];

     end

% Feature Vector to be classified

      if (C==2)

  fprintf('\n Insert the feacture vector to be classified. \n ')


            if (M==1)

                 yax=linspace(-0.05,0.05,10);

                 xsample_1=input('Insert the Feature:');

                 xsample=[1;xsample_1];
```

```
figure(C+1)
plot(xsample_1,0,'go');
% Calculating Vector of Weights
W1=Pocket(X,y_ort,M,N,C);
w_ini=(rand(1,M+1))' ;% Initialization Vector
W2 =perce(X,y_ort,M,N,C,w_ini);
W3= LMS(X,M,N,C,W2,y_ort );
W4=SOE(X,M,N,C) ;
w01=W1(1);
w11=W1(2);
w02=W2(1);
w12=W2(2);
w03=W3(1);
w13=W3(2);
w04=W4(1);
w14=W4(2);
% With Pocket
cd(FolderName)
hgload('1Feature')
hold on
figure(C+2)
plot(xsample_1,0,'go');
hold on
plot((-w01/w11)*ones(1,10),yax,'-.k*')
hold on
title('Discriminant Function with Pocket Algorithm');
hgsave('WithPocket')
cd ..
% With Perceptron
cd(FolderName)
hgload('1Feature')
hold on

figure(C+3)
```

```
                    plot(xsample_1,0,'go');
                    hold on
                    plot((-w02/w12)*ones(1,10),yax,'-.k*')
                    hold on
                    title('Discriminant Function with Perceptron
Algorithm');
                    hgsave('WithPercpetron')
                    cd ..
                    %With LMS
                    cd(FolderName)
                    hgload('1Feature')
                    hold on
                    figure(C+4)
                    plot(xsample_1,0,'go');
                    hold on
%
                    plot((-w03/w13)*ones(1,10),yax,'-.k*')
                    hold on
                    title('Discriminant Function with LMS Algorithm');
                    hgsave('WithLMS')
                    cd ..
                    %With SOE
                    cd(FolderName)
                    hgload('1Feature')
                    hold on

                    figure(C+5)
                    plot(xsample_1,0,'go');
                    hold on

                    plot((-w04/w14)*ones(1,10),yax,'-.k*')
                    hold on
                    title('Discriminant Function with SOE Algorithm');
                    hgsave('WithSOE')
                    cd ..
```

```
            elseif(M==2)

                xsample_1= input('Insert Feature 1:');

                xsample_2= input('Insert Feature 2:');

                xsample=[1 ;xsample_1; xsample_2];

                figure(C+1)

                plot(xsample(2),xsample(3),'go');

                hold on

                xax= linspace((min_values(1))-0.5,max_values(1)+0.5,1000);


        % Calculating Vector of Weights

            % With Pocket

                W1=Pocket(X,y_ort,M,N,C);

                w01=W1(1);

                w11=W1(2);

                w21=W1(3);

                y1= -(w01+(w11*xax))/w21;

                cd(FolderName);


                hgload('2Features');

                hold on;

                figure(C+2)

                plot(xax,y1,':k','LineWidth',2);

                hold on;

                figure(C+2)

                plot(xsample(2),xsample(3),'go');

                title('Discriminant Function with Pocket Algorithm');

                %axis([ (min_values(1)-2) (max_values(1)+2)  (min_values(2)-
2) (max_values(2)+2)])

                hold on;

                hgsave('WithPocket');

                cd ..


            %With Perceptron
```

```
w_ini=(ones(1,M+1))' ;% Initialization Vector
W2 =perce(X,y_ort,M,N,C,w_ini);
w02=W2(1);
w12=W2(2);
w22=W2(3);
y2= -(w02+(w12*xax))/w22;
cd(FolderName);
hgload('2Features');
figure(C+3)
plot(xax,y2,':k','LineWidth',2);
hold on;
plot(xsample(2),xsample(3),'go');
hold on;
title('Discriminant Function with Perceptron Algorithm');
hold on;
hgsave('WithPerceptron');
cd ..

%With LMS
W3= LMS(X,M,N,C,W1,y_ort );
w03=W3(1);
w13=W3(2);
w23=W3(3);
y3= -(w03+(w13*xax))/w23;
cd(FolderName);
hgload('2Features');
figure(C+4)
plot(xax,y3,':k','LineWidth',2);
hold on;
plot(xsample(2),xsample(3),'go');
hold on;
title('Discriminant Function with LMS Algorithm');
hold on;
```

```
    hgsave('With LMS');
    cd ..


%With SOE
    W4=SOE(X,M,N,C) ;
    w04=W4(1);
    w14=W4(2);
    w24=W4(3);
    y4= -(w04+(w14*xax))/w24;
    cd(FolderName);
    hgload('2Features');
    figure(C+5)
    plot(xax,y4,':k','LineWidth',2);
    hold on;
    plot(xsample(2),xsample(3),'go');
    hold on;
    title('Discriminant Function with SOE Algorithm');
    hold on ;
    hgsave('WithSOE');
    cd ..


   elseif(M==3)

    xsample_1= input('Insert Feature 1:')
    xsample_2= input('Insert Feature 2:')
    xsample_3=input('Insert Feature 3:')
    xsample=[1 ;xsample_1; xsample_2; xsample_3];
    plot3(xsample(2),xsample(3),xsample(4),'go');
    hold on;
    xaxis= linspace((min_values(1))-0.5,max_values(1)+0.5,100);
    yaxis= linspace(min_values(2)-0.5,max_values(2)+0.5,100);
     % With Pocket
```

```
W1=Pocket(X,y_ort,M,N,C);
w01=W1(1);
w11=W1(2);
w21=W1(3);
w31=W1(4);
z1=zeros(100,100);
for t=1:100
    for u=1:100
        z1(t,u)=(-1)*(w01+w11*xaxis(u)+w21*yaxis(t))/w31;
    end
end
cd(FolderName);
hgload('3Features');
hold on
figure(C+2)
plot3(xsample(2),xsample(3),xsample(4),'go');
hold on;
mesh(xaxis,yaxis,z1);
hold on;
title('Discriminant Function with Pocket Algorithm');
hgsave('WithPocket');
cd ..
%With Percetron
w_ini=(rand(1,M+1))' ;% Initialization Vector
W2 =perce(X,y_ort,M,N,C,w_ini);
w02=W2(1);
w12=W2(2);
w22=W2(3);
w32=W2(4);
z2=zeros(100,100);
for t=1:100
    for u=1:100
        z2(t,u)=(-1)*(w02+w12*xaxis(u)+w22*yaxis(t))/w32;
```

```
    end
end
cd(FolderName);
hgload('3Features')
hold on;
figure(C+3)
plot3(xsample(2),xsample(3),xsample(4),'go');
hold on;
mesh(xaxis,yaxis,z2);
hold on;
title('Discriminant Function with Perceptron Algorithm')
hgsave('WithPerceptron');
cd ..

%With LMS
W3= LMS(X,M,N,C,W2,y_ort );
w03=W3(1);
w13=W3(2);
w23=W3(3);
w33=W3(4);
z3=zeros(100,100);
for t=1:100
    for u=1:100
        z3(t,u)=(-1)*(w03+w13*xaxis(u)+w23*yaxis(t))/w33;
    end
end
cd(FolderName);
hgload('3Features');
hold on;
figure(C+4)
plot3(xsample(2),xsample(3),xsample(4),'go');
hold on;
mesh(xaxis,yaxis,z3);
hold on;
```

```
            title('Discriminant Function with LMS Algorithm')
            hgsave('WithLMS')
            cd ..

        %With SOE
            W4=SOE(X,M,N,C) ;
            w04=W4(1);
            w14=W4(2);
            w24=W4(3);
            w34=W4(4);
            z4=zeros(100,100);
            for t=1:100
                for u=1:100
                    z4(t,u)=(-1)*(w04+w14*xaxis(u)+w24*yaxis(t))/w34;
                end
            end
            cd(FolderName);
            hgload('3Features');
            hold on;
            figure(C+5)
            plot3(xsample(2),xsample(3),xsample(4),'go');
            hold on;
            mesh(xaxis,yaxis,z4);
            hold on;
            title('Discriminant Function with SOE Algorithm')
            hgsave('WithSOE')
            cd ..

    else
        xsample1= input('Insert the Vector = ')
        xsample=[1;xsample1(1:M,1)];
        W1=Pocket(X,y_ort,M,N,C);
        w_ini=(rand(1,M+1))' ;% Initialization Vector
        W2 =perce(X,y_ort,M,N,C,w_ini);
```

```
            W3= LMS(X,M,N,C,W2,y_ort );

            W4=SOE(X,M,N,C) ;


        end

            % Discrimination Stage for 2_Class case

            Threshold1=W1'*xsample;


                 if Threshold1<0

                        fprintf('\n Second Pocket Algorithm the Vector takes
part of the  Class 2  \n');
                 else

                         fprintf('\n Second Pocket Algorithm the Vector
takes part of the  Class 1 \n');

                 end


            Threshold2=W2'*xsample;


                 if Threshold2<0

                        fprintf('\n Second Pocket Algorithm the Vector takes
part of the  Class 2  \n');
                 else

                         fprintf('\n Second Pocket Algorithm the Vector
takes part of the  Class 1 \n');

                 end

            Threshold3=W3'*xsample;


              if Threshold3<0

                        fprintf('\n Second Perceptron Algorithm the Vector
takes part of the  Class 2  \n');
                 else

                         fprintf('\n Second Perceptron Algorithm the Vector
takes part of the  Class 1 \n');

                 end


            Threshold4=W4'*xsample;
```

```
                if Threshold4<0

                    fprintf('\n Second LMS Algorithm the Vector takes
part of the  Class 2  \n');

                else

                    fprintf('\n Second LMS Algorithm the Vector takes
part of the  Class 1 \n');

                end


            % C_Class Classifier (C>2)

        else

            xax= linspace((min_values(1))-0.5,max_values(1)+0.5,100);


            fprintf('\n Insert the feacture vector to be classified. \n ')

             if (M==1)

                    xsample_1=input('Insert the Feature:');

                    xsample=[xsample_1;1];

                    plot(xsample_1,0,'ko');

                    hold on

             elseif(M==2)


                xsample_1= input('Insert Feature 1:')

                xsample_2= input('Insert Feature 2:')

                xsample=[xsample_1; xsample_2;1];

                plot(xsample(1),xsample(2),'ko')

             elseif(M==3)

                yax= linspace((min_values(2))-0.5,max_values(2)+0.5,1000);

                xsample_1= input('Insert Feature 1:')

                xsample_2= input('Insert Feature 2:')

                xsample_3=input('Insert Feature 3:')

                xsample=[xsample_1; xsample_2; xsample_3;1];

                plot3(xsample(1),xsample(2),xsample(3),'ko')
```

```
            else
            xsample1= input('Insert the  Vector = ')
            xsample=[xsample1(1:M,1);1];


        end

            Matrix_Kesler=Kesler_Construc(X,M,N,C);
            w1_Cclass=perceMultiClass(Matrix_Kesler);
            fprintf('\n Second Perceptron Algorithm :\n')
            discriminant_Cclass(xsample,C,M,w1_Cclass);
            w2_Cclass= PocketMultiClass(Matrix_Kesler);
            fprintf('\n Second Pocket Algorithm :\n')
            discriminant_Cclass(xsample,C,M,w2_Cclass);

            %---------------------------------------
            matrix_weightLMS=zeros(M+1,C);

            w_ini=(rand(1,M+1))' ;% Initialization Vector


            for o=1:C
                matrix_weightLMS(1:M+1,o)=LMSMultiClass(X,M,N,C,w_ini,o );
           end


            %Discrimant Block
            ThersholdLMSMultiClass=zeros(1,C);
            for g=1:C
                ThersholdLMSMultiClass(g)=
(matrix_weightLMS(1:M+1,g))'*xsample;
            end

ClassLMS=find(ThersholdLMSMultiClass==max(ThersholdLMSMultiClass));
            fprintf('\n Second LMS Algorithm the vector takes part of the
Class %d \n',ClassLMS)
            %------------------------------------------------------------
```

```
            matrix_weightSOE=zeros(M+1,C);

            for r=1:C

                matrix_weightSOE(1:M+1,r)= SOEMultiClass(X,M,N,C,r) ;

            end

            %Discrimant Block

            ThersholdSOEMultiClass=zeros(1,C);

            for u=1:C

                ThersholdSOEMultiClass(u)=
(matrix_weightSOE(1:M+1,u))'*xsample;

            end

ClassSOE=find(ThersholdSOEMultiClass==max(ThersholdSOEMultiClass));

            fprintf('\n Second SOE Algorithm the vector takes part of the
Class %d \n',ClassSOE)

     end
```

2.cbuild

```
% This function creates a variable folder name

function FolderName=cbuild(C,M)

FolderName='';

FolderName=[FolderName num2str(C) 'Classes' '_' num2str(M) 'Features' '_'
date '_' num2str(cputime)];
```

3.Perceptron

```
% Perceptrom Algorithm

function w =perce(X,y,M,N,C,w_ini)

Ntotal=N*C;

Xper=[ones(1,Ntotal); X(1:M,1:Ntotal)];


[m,Ntotal]=size(Xper);

yper=y;

max_iter=100000; % Maximum allowable number of iterations
```

```
rho=1; % Learning rate

w=w_ini; % Initialization of the parameter vector

iter=0; % Iteration counter

mis_clas=Ntotal; % Number of misclassified vectors

while(mis_clas>0)&&(iter<max_iter)


    iter=iter+1;

    mis_clas=0;

    gradi=zeros(M+1,1) ;% Inizialitation of the gradient term

    for i=1:Ntotal

        if((Xper(:,i)'*w)*yper(i)<0) % Verification Perceptron cost

            mis_clas=mis_clas+1;

            gradi=gradi-(yper(i)*Xper(:,i)); % Computation of the gradient
term
        end

    end

    w=w-(1/iter)*gradi; %Updating the parameter vector
end

fprintf('\n Iteration Number with Perceptron %d \n',iter);
```

4.Pocket

```
% Pocket Algorithm with Ratchet

function W= Pocket(X,y,M,N,C)

%X=[] Training Feature Vectors Matrix

%y=[] Vector of desired responses

%W=[] Vector of integral pocket weights

%pi=[] Vector of integral perceptron weights

%run_pi= number of consecutive correct classifications using perceptron
weights pi

%run_w= number of consecutive correct classifications using pockets weights
W

%num_okpi= total number of training examples that pi correctly classifies.

%num_okw= total number of training examples that W correctly classifies.

Ntotal=N*C;

Xpocket=[ones(1,Ntotal); X(1:M,1:Ntotal)];
```

```
[m,Ntotal]=size(Xpocket);

pi=(zeros(1,m))';
run_pi=0;
run_w=0;
num_okpi=0;
num_okw=0;
num_Iteration=100000;
Iteration_counter=1;
index= ceil(Ntotal*rand(1,1)) ;% Randomly pick a training example
x_sample= Xpocket(1:m,index);
y_sample= y(index);


        while ((Iteration_counter<num_Iteration) && (num_okw<Ntotal))


                f_sample=pi'*x_sample;
                    if (((f_sample >0) && (y_sample ==1)) || ((f_sample <0)
&& (y_sample ==-1)))


                                run_pi=run_pi+1;
                                    if (run_pi>run_w)
                                %Compute num_okpi by checking every training
example
                                 for i=1:Ntotal
                                   f_sample_vector(i)=pi'*Xpocket(1:m,i);
                                 end
                                 thershold=f_sample_vector.*y;
                                 correctly_index=find(thershold>0);
                                 num_okpi=length(correctly_index);
                                    if (num_okpi>num_okw)
                                        W=pi;
                                        run_w=run_pi;
                                        num_okw=num_okpi;
```

```
                                    if (num_okw==Ntotal)
                                    break

                                    end
                                end
                                end
                        index= ceil(Ntotal*rand(1,1)); % Randomly pick a
training example
                        x_sample= Xpocket(1:m,index);
                        y_sample= y(index);
                        f_sample=pi'*x_sample;




                    else

                            pi=pi+y_sample*x_sample;
                            run_w=0;
                            run_pi=0;



                        end
                Iteration_counter=Iteration_counter+1;
            end
            fprintf('\n Iteration Number with Pocket: %d \n',Iteration_counter);
```

5.LMS

```
% LMS Algorithm
function w= LMS(X,M,N,C,W2,y )
Ntotal=N*C;
XLMS=[ones(1,Ntotal); X(1:M,1:Ntotal)];
% y=zeros(1,Ntotal);
```

```
%
% for t=1:Ntotal
%
%     y(t)=W2'*XLMS(1:(M+1),t);
%
% end


[m,Ntotal]=size(XLMS);


w=W2;


rhoK= 4e-10; % Learning rate

for i=1:Ntotal

    w= w+((rhoK)*(y(i)-(XLMS(1:m,i))'*w)*XLMS(1:m,i));
end
```

6.SOE

```
%Sum of Error Squares Estimation



function w= SOE(X,M,N,C)

% Ntotal=N*C;
%
% XSOE=[ones(1,Ntotal); X(1:M,1:Ntotal)];
%
%
% [m,Ntotal]=size(XSOE);
%
% y=zeros(1,Ntotal)';
%
% for t=1:Ntotal
%
%     y(t)=W2'*XSOE(1:(M+1),t);
%
% end
%
% W=(zeros(1,m))';
%
%
%
% %Compute the weights vector
%
%
%      W= inv(XSOE*XSOE')*(XSOE*y);
%

Ntotal=N*C;

XSOEinitial=[ones(1,Ntotal); X(1:M,1:Ntotal)];

Y1=XSOEinitial';

Y=[Y1(1:N,1:M+1);(-1)*Y1(N+1:2*N,1:M+1)];
```

```
w=ones(1,M+1)';

y=ones(Ntotal,1);

rho=0.9;

% bmin=0.01;

MaxIteration=1500000;


for k=1:MaxIteration

    error=(Y*w)-y;

    errorpositive=(error+abs(error))/2;

    y=y+2*(rho)*errorpositive;

    w=(inv(Y'*Y)*Y')*y;

    if(Y*w >0)

        break;

    end

end


    fprintf('K=,%d',k);
%
%     x=linspace(1,100,1000);
%
%     w1=w(1);
%
%     w2=w(2);
%
%     w3=w(3);
%
%     ynuevo =zeros(1,1000);
%
%     ynuevo =(-1)*((w2*x)+w1)/w3;
%
%     plot(x,ynuevo)
%     hold on;
%     grid on;
%
%     axis([1 50 0 20]);
%
```

7.Kesler

```
% M Class Case Classification

function Matrix_Kesler=Kesler_Construc(X,M,N,C)



Ntotal=N*C;
```

```
X1= [X(1:M,1:Ntotal); ones(1,Ntotal)];

[m,Ntotal]=size(X1);

Xtraining_extension=zeros(m*C,C*N);

Xtotale=zeros(m*C,C*N);

for i=1:C

    for p=1:N

        for j=1:C

            if (i==j)


            Xtraining_extension(1:m*C,(N*C*(i-1))+(j+C*(p-1)))=
zeros(1:m*C,1);


            else

             Xtraining_extension(((i-1)*m)+1:((i-1)*m)+m,(N*C*(i-
1))+(j+C*(p-1)))= X1(1:m,p+N*(i-1));

            Xtraining_extension(((j-1)*m)+1:((j-1)*m)+m,(N*C*(i-1))+(j+C*(p-
1)))= (-1)*X1(1:m,p+N*(i-1));

            end


        end

    end

    %Xtotale (1:m*C,1+N*C*(i-1):N*C*(i-
1)+(N*C))=Xtraining_extension(1:m*C,1:C*N)
end

f=Xtraining_extension;

[a,b]=size(f);

index=any(f);

index_def=zeros(1,length(index));

for i=1:length(index)

    if(index(i)==1)

        index_def(i)=i;

    end

end

t=0;

vect_f=zeros(1,Ntotal*(C-1));

x=length(vect_f);

for j=1:length(index_def)
```

```
    if(index_def(j)~=0)
        t=t+1;
        vect_f(t)=index_def(j);
    end
end
%Final Kesler Construction
Matrix_Kesler= zeros(a,x);
for k=1:x
    Matrix_Kesler(1:a,k)=(f(1:a,vect_f(k)));
end
```

8.Perceptron multiclass

```
% Perceptron Algorithm for Multiclass Case
function w=perceMultiClass(Matrix_Kesler)
[o,p]=size(Matrix_Kesler);
y=ones(1,p);
yper=y;
max_iter=100000; % Maximum allowable number of iterations
rho=1; % Learning rate

% Initialization of the parameter vector


w=(zeros(1,o))';
for h=1:o
    w(h)= rand(1,1);
end
iter=0; % Iteration counter
mis_clas=p; % Number of misclassified vectors
while(mis_clas>0)&&(iter<max_iter)
    iter=iter+1;
    mis_clas=0;
    gradi=zeros(o,1) ;% Inizialitation of the gradient term
    for i=1:p
        if((w'*Matrix_Kesler(:,i))*yper(i)<0) % Verification Perceptron cost
```

```
            mis_clas=mis_clas+1;

            gradi=gradi-(yper(i)*Matrix_Kesler(:,i)); % Computation of the
gradient term
        end

    end

    w=w-(rho/iter)*gradi; %Updating the parameter vector
end

fprintf('Iteration Counter %d',iter);
```

9. Pocket Multi-Class

```
% Pocket Algorithm with Ratchet for Multi-Class Case

function W= PocketMultiClass(Matrix_Kesler)

% Pocket Algorithm with Ratchet


%X=[] Training Feature Vectors Matrix

%y=[] Vector of desired responses

%W=[] Vector of integral pocket weights

%pi=[] Vector of integral perceptron weights

%run_pi= number of consecutive correct classifications using perceptron
weights pi

%run_w= number of consecutive correct classifications using pockets weights
W

%num_okpi= total number of training examples that pi correctly classifies.

%num_okw= total number of training examples that W correctly classifies.

Xpocket=Matrix_Kesler;

[u,v]=size(Xpocket);

y=(ones(1,v));

pi=(zeros(1,u))';

run_pi=0;

run_w=0;

num_okpi=0;

num_okw=0;

num_Iteration=100000;

Iteration_counter=1;

index= ceil(v*rand(1,1)) ;% Randomly pick a training example

x_sample= Xpocket(1:u,index);
```

```
y_sample= y(index);
f_sample_vector= zeros(1,v);
thershold=zeros(1,v);


        while ((Iteration_counter<num_Iteration) && (num_okw<v))
                f_sample=pi'*x_sample;
                    if (f_sample >0)

                                run_pi=run_pi+1;
                                if (run_pi>run_w)
                               %Compute num_okpi by checking every training
example
                                for i=1:v
                                  f_sample_vector(i)=pi'*Xpocket(1:u,i);
                                end
                                thershold=f_sample_vector.*y;
                                correctly_index=find(thershold>0);
                                num_okpi=length(correctly_index);
                                   if (num_okpi>num_okw)
                                       W=pi;
                                       run_w=run_pi;
                                       num_okw=num_okpi;
                                           if (num_okw==v)
                                           break

                                           end
                                   end
                                   end
                         index= ceil(v*rand(1,1)); % Randomly pick a
training example
                        x_sample= Xpocket(1:u,index);
                        y_sample= y(index);
                        f_sample=pi'*x_sample;
```

```
                else

                        pi=pi+y_sample*x_sample;
                        run_w=0;
                        run_pi=0;



                    end
            Iteration_counter=Iteration_counter+1;
        end
        fprintf('\n Iteration Number with Pocket: %d \n',Iteration_counter);
```

10.LMS Multi-Class

```
% LMS Algorithm
function w= LMSMultiClass(X,M,N,C,w_ini,o )
Ntotal=N*C;

XLMS=[X(1:M,1:Ntotal);ones(1,Ntotal)];

[m,Ntotal]=size(XLMS);
y=zeros(1,Ntotal);
y(1+N*(o-1):o*N)=ones(1,N);

w=w_ini;

rhoK= 0.1; % Learning rate
for i=1:Ntotal
    w= w+((rhoK/i)*(y(i)-(XLMS(1:m,i))'*w)*XLMS(1:m,i));
end
```

11. Discriminant Block

```
function discriminant_Cclass(xsample,C,M,w)
m=M+1;
Xsamplematrix=zeros(m*C,C*C);
[s,t]=size(Xsamplematrix);
for q=1:C

   for h=1:C
       if (h==q)
         Xsamplematrix(1:s,(h+C*(q-1)))= zeros(s,1);

       else
       Xsamplematrix((q-1)*m+1:((q-1)*m)+m,(h+C*(q-1)))= xsample(1:m,1);
       Xsamplematrix((h-1)*m+1:((h-1)*m)+m,h+C*(q-1))= (-1)*xsample(1:m,1);
       end

   end
end
matrixdef=Xsamplematrix;
indexsample=any(Xsamplematrix);
index_defsample=zeros(1,length(indexsample));
for i=1:length(indexsample)
    if(indexsample(i)==1)
        index_defsample(i)=i;
    end
end
b=0;
vect_fsample=zeros(1,C*(C-1));
xs=length(vect_fsample);
for u=1:length(index_defsample)
    if(index_defsample(u)~=0)
        b=b+1;
        vect_fsample(b)=index_defsample(u);
    end
end
%Final Kesler Construction
```

```
Matrix_KeslerSample= zeros(s,xs);
for g=1:xs
    Matrix_KeslerSample(1:s,g)=(Xsamplematrix(1:s,vect_fsample(g)));
end


            %Discrimination
Thershold=zeros(C-1,C);
for v=1:C
   for y=1:C-1
    Thershold(y,v)=w'*Matrix_KeslerSample(1:s,y+(C-1)*(v-1));
   end
end
min1=min(Thershold);
for wclass=1:C;
    minth=min1(wclass);
     if (minth>0)
        fprintf('\n The vector takes part of the Class %d \n ',wclass)
    end
end
        end


% Thershold=zeros(1,C)
%
% for g=1:C
%
%   Thershold(g)=w(1+m*(g-1):g*m,1)'*xsample
%
% end
%
% index_class=find(Thershold==max(Thershold))
```

Wifi-Bluetooth Classifier

1.Script

```
%Script

close all;
clear;
clc;
C=2;
M=2;
```

```
l=1000;

%Bluetooth traffic Generation (Classifier Input)

  %[DurationVector,TSFVector]=MSBluetoothTrafficGenerationsample(l);

  [DurationVector,TSFVector]=SSBluetoothTrafficGenerationsample(l);



%  %Wi-Fi traffic Generation (Classifier Input)

%        filename = 'VAIOHPATHSKYPEUTUBEOPENOFFICEINFOCOM.txt';
%
%        [TSFVector,DurationVector]=import_WiFi(filename) ;
%
% % %
% Mixed Wi-Fi Bluetooth Input
%[DurationVector,TSFVector]=MixedMSBluetoothWiFi;

%Feature Extraction

[IFSVector,MaxPPDUduration,Interference] =
FeatureExtraction(DurationVector,TSFVector);

L=length(IFSVector);

 %Folder Name's Construction

 FolderName=cbuild(C,M);

 mkdir( FolderName);


%Classification
[xax,y1,y2,y3,y4,W1,W2,W3,W4]= Classifiers(FolderName);




%

% %Multi-Slot
% W1=[27;-2951.00750966092;214];
%
% W2=[-20.9780909571695;-754754.944960720;65177.3137731307];
%
% W3=[-2.60459375983088e+40;-1.91568366881445e+40;3.80307560277537e+39];
%
% W4=[2885.28906276758;-12.2577405322002;0.213666512014766];


%SingleSlot
% W1=[-3;-1160.13813388255;614];
%
% W2=[-1124.66666666667;-596979.329122128;393714.500000000];
% W3=[-791.939264646996;-412350.058961372;241720.118250060];
% W4=[-767.507895440477;-6.77765685180857;6.59157513268478];
% %


%SingleSlot N=10000
```

```
% W1=[-11; -1951.51989846758 ; 1254];
% W2=[-729.199999999998;-296965.142763335;143747.200000000];
% W3=[-11.0557264886324;-1950.94146912034; 1160.08983006525];
% W4=[-780.131200548504; -6.91890575158495;6.67045354238516];

%Multi Slot N=10000

% W1=[114;-7400.15315093904;619];
% W2=[438966.699999974;-894011.013611800;52211.6000000255];
% W3=[124.716346120036;-3852.40422992936;744.923713217399 ];
% W4=[1492.39002650608; -17.9499939253218;1.49834503999549];


%
%
%
%
% figure (8)
%
%                 plot(xax,y1,':k','LineWidth',2);
%
%                 title('Discriminant Function with Pocket Algorithm');
%
%                 axis([ 1 700 1 5000]);
%
%                 hold on;
%
%
%                 for a=1:L
%
%                     plot(IFSVector(a),MaxPPDUduration(a),'*m')
%
%                     hold on
%
%                 end
%
%                 grid on
%
%                 xlabel('Duration of Silent Gaps (usec)')
%
%                 ylabel('Max PPDU Duration between two Silent Gaps (usec)')
%
%                 cd(FolderName);
%
%                 hgsave('Sample_Pocket');
%
%                 cd ..
%
%
% figure (9)
%
%                 plot(xax,y2,':k','LineWidth',2);
%
%                 title('Discriminant Function with Perceptron Algorithm');
%
%                 axis([ 1 700 1 5000]);
%
%                 hold on;
%
%                  for b=1:L
%
%                     plot(IFSVector(b),MaxPPDUduration(b),'*m')
%
%                     hold on
%
%                  end
```

```
%
%                   grid on
%
%                  xlabel('Duration of Silent Gaps (usec)')
%
%                  ylabel('Max PPDU Duration between two Silent Gaps (usec)')
%
%                  cd(FolderName);
%
%                  hgsave('Sample_Perceptron');
%
%                  cd ..
%
%
%
%
%
% figure (2)
%
%                  plot(xax,y3,':k','LineWidth',2);
%
%                  title('Discriminant Function with LMS Algorithm');
%
%                  axis([ 1 700 1 5000]);
%
%                  xlabel('Inter Frame Space (usec)');
%
%                  ylabel('Max PPDU Duration (usec)');
%
%                  hold on;
%
%
%                   for c=1:L
%
%
%
%                      plot(IFSVector(c),MaxPPDUduration(c),'*m')
%
%                      hold on
%
%                   end
%
%                  grid on
%
%                  xlabel('Duration of Silent Gaps (usec)')
%
%                  ylabel('Max PPDU Duration between two Silent Gaps (usec)')
%
%                  cd(FolderName);
%
%                  hgsave('Sample_LMS');
%
%                  cd ..
%
%
%
%
% figure (3)
%
%                  plot(xax,y4,':k','LineWidth',2);
%
%                  title('Discriminant Function with SOE Algorithm');
%
%                  xlabel('Inter Frame Space (usec)');
%
%                  ylabel('Max PPDU Duration (usec)');
```

```
%
%                  axis([ 1 700 1 5000]);
%
%                  hold on;
%
%                   for d=1:L
%
%                      plot(IFSVector(d),MaxPPDUduration(d),'*m')
%
%                      hold on
%
%                   end
%
%                     grid on
%
%                  xlabel('Duration of Silent Gaps (usec)')
%
%                  ylabel('Max PPDU Duration between two Silent Gaps (usec)')
%
%                  cd(FolderName);
%
%                  hgsave('Sample_SOE');
%
%                  cd ..
%
%
cd (FolderName)

hgload('2Features')

hold on

 figure(3)

                plot(xax,y1,'r');

                 hold on;

                  plot(xax,y2,'k');

                hold on;

                    plot(xax,y3,'g');

               hold on;


                   plot(xax,y4,'m');

                hold on;


                hgsave('todo')

                 cd ..

%
%
%


                 %Discriminant Block

                 j
=DiscriminantBlock(W1,W2,W3,W4,L,IFSVector,MaxPPDUduration)
```

2.cell2str

```
function str = cell2str(c)
% Convert a cell array of strings into an array of strings.
% CELL2STR pads each string in order to force all strings
% have the same length.
%

% Determine the length of each string in cell array c
nblanks = cellfun(@length, c);
maxn = max(nblanks);
nblanks = maxn-nblanks;

% Create a cell array of blanks.  Each column of the cell array contains
% the number of blanks necessary to pad each row of the converted string
padding = cellfun(@blanks,num2cell(nblanks), 'UniformOutput', false);

% Concatinate cell array and padding
str = {c{:}; padding{:}};

% This operation converts new the cell array into a string
str = [str{:}];

% Reshape the string into an array of strings
ncols = maxn;
nrows = length(str)/ncols;
str = reshape(str,ncols,nrows)';
```

3.Classifier

```
%Classifier Script

function [xax,y1,y2,y3,y4,W1,W2,W3,W4]= Classifiers(FolderName)


close all;

C=2;

M=2;



%Wi_Fi Characterization


[TSFVectorWIFI,DurationVectorWIFI]=WIFITrainingGeneration;



[IFSVectorWIFI,MaxPPDUdurationWIFI,InterferenceWIFI]=FeatureExtraction(Durat
ionVectorWIFI,TSFVectorWIFI);


% Bluetooth Charaterization

[DurationVectorBluetooth,TSFVectorBluetooth]=SSBluetoothTrafficGeneration;

%[DurationVectorBluetooth,TSFVectorBluetooth]=MSBluetoothTrafficGeneration;
```

```
[IFSVectorBluetooth,MaxPPDUdurationBluetooth,InterferenceBluetooth]=FeatureE
xtraction(DurationVectorBluetooth,TSFVectorBluetooth);


%Calculating the Feauture's Histogram

Feature_matrix=zeros(length(IFSVectorWIFI),C*M);









              Feature_matrix(1:length(IFSVectorWIFI),1)= IFSVectorWIFI; %IFS
Wi-Fi


              Feature_matrix(1:length(IFSVectorWIFI),2)=
MaxPPDUdurationWIFI; %Max-Duration Wi-Fi


Feature_matrix(1:length(IFSVectorWIFI),3)=IFSVectorBluetooth(1:length(IFSVec
torWIFI)) ; % IFS Bluetooth


Feature_matrix(1:length(IFSVectorWIFI),4)=MaxPPDUdurationBluetooth(1:length(
IFSVectorWIFI)); %Max-Duration Bluetooth


              X1=Feature_matrix;


  [N,columna]=size(Feature_matrix) % Training Matrix

   %Folder Name's Construction

%     % Plotting Histograms
%
%
%
%
%              figure(1) %(Bluetooth)
%
%              subplot(2,1,1),hist(Feature_matrix(1:N,3),0:1:625);
%
%              title('Characterization of Silence Gaps Duration
(Bluetooth)');
%
%              grid on;
%
%              hold on;
%
%              axis([0 700 0 2000]);
%
%              subplot(2,1,2),hist(Feature_matrix(1:N,4),0:100:3000);
%
%              title('Characterization of Max PPDU Duration between two
```

```
Silence Gaps  (Bluetooth)');
%
%               grid on;
%
%               hold on;
%
%               axis([0 4000 0 3000]);
%
%               cd(FolderName);
%
%               hgsave('BluetoothCharacterization')
%
%               cd ..
%


%               figure(2) %(Wi-Fi)
%
%
%
%               subplot(2,1,1),hist(Feature_matrix(1:N,1),0:1:625);
%
%               title('Characterization of Silence Gaps Duration
(802.11b/g)');
%
%               grid on;
%
%               hold on;
%
%               axis([0 700 0 2000]);
%
%
subplot(2,1,2),hist(Feature_matrix(1:N,2),0:100:3000,'FaceColor',[0.03922
0.1412 0.4157]);
%
%               grid on;
%
%               title('Characterization of Max PPDU Duration between two
Silence Gaps  (802.11b/g)');
%
%               hold on
%
%               axis([0 4000 0 3000])
%
%               cd(FolderName);
%
%               hgsave('Wi_FiCharacterization')
%
%               cd ..
%
%


%    Plotting the M-dimensional Space


    ColorSet=[1 0 0 ; 0 0 1 ; 0 1 0 ; 0 1 1 ; 1 0 1 ;0 0 0;1 1 0];

    controlC=C-1;
```

```
                            figure(1)

%                             for f=1:M:M*C
%
%
p2=plot(X1(1:N,f),X1(1:N,f+1),'+','Color',ColorSet(1+((f-1)/M),:));
%
%                                 hold on
%
%                             end

%Paper-----------------------------------------------------------------------

p1=plot(X1(1:N,1),X1(1:N,2),'+','Color',ColorSet(1,:));
                            hold on

p2=plot(X1(1:N,3),X1(1:N,4),'o','Color',ColorSet(2,:));
                            hold on
                            %_____________________________________________

                            title('FEATURES  SPACE');
                            legend('Wi-Fi Training','Bluetooth Training',2);
                            axis([ 1 700 1 5000])
                            xlabel('Duration of Silence Gaps [?sec]')
                            ylabel('Max Packet Duration between two Silence
Gaps [?sec]')
                            grid on
                            cd(FolderName)
                            hgsave('2Features')
                        cd ..
                        figure(2)
                        Xp= [];
                        Xp(1:N,1) = X1(1:N,1);
                        Xp(1:N,2) = X1(1:N,2);
                        hist3(Xp, {0:5:625
0:100:3000},'FaceAlpha',.65,'LineStyle','none','FaceColor',[1 0 0]);
                        title('Data Point Density Histogram and Intensity
Map');
                        grid on
                        view(3);
                        xlabel('Duration of Silence Gaps');
                        ylabel('Max Packet Duration between two Silence
Gaps');
                        zlabel('Ocorrence');
                        set(gcf,'renderer','opengl');
                        hold on
                        figure(2)
```

```
                        Xp= [];
                        Xp(1:N,1) = X1(1:N,3);
                        Xp(1:N,2) = X1(1:N,4);
                        hist3(Xp, {0:5:625
0:100:3000},'FaceAlpha',.65,'LineStyle','none','FaceColor',[ 0 0 1]);
                        title('Data Point Density Histogram and Intensity
Map');
                        grid on
                        view(3);
                        xlabel('Duration of Silence Gaps');
                        ylabel('Max Packet Duration between two Silence
Gaps');
                        zlabel('Ocorrence');
                        set(gcf,'renderer','opengl');
                        hold on
```

```
%    %Generation matrix of Training (M,N*C) dimension
%

    D=N*C;% Total Number of Training


    Xp=zeros(M,N); %Training Matrix

    X_convertion= (X1)';

    Xq=zeros(M,N*C);

    X=zeros(1:M,N*C);



    for k=1:C

         Xp(1:M,1:N)= X_convertion(1+M*(k-1):(k*M),1:N);

         Xq=[Xq Xp];

         Xp=zeros(M,N);

    end

  X=Xq(1:M,(N*C)+1:2*(N*C));
%
% %  Axis Construction
%
  max_values=max(X,[],2);

  min_values=min(X,[],2);


  %Vector of Desired Response Generation
```

```
        y_ort=[ones(1,N) (ones(1,N)*(-1))];

        xax= linspace(-100,700,1000);

%
%          %Calculating Vector of Weights
%
%
%
%
%             %With Pocket

                W1=Pocket(X,y_ort,M,N,C);

                w01=W1(1);

                w11=W1(2);

                w21=W1(3);

                y1= -(w01+(w11*xax))/w21;

                cd(FolderName);

                hgload('2Features');

                hold on;

                figure(C+2)

                plot(xax,y1,':k','LineWidth',2);

                hold on;

                %plot(xsample(2),xsample(3),'go');

                title('Discriminant Function with Pocket Algorithm');

                axis([ 1 700 1 5000])

                corte1=-w01/w21;

                fprintf('Corte %f',corte1)

                hold on;

                hgsave('WithPocket');

                cd ..

           % With Perceptron

                w_ini=ones(1,M+1)' ;% Initialization Vector

                W2 =perce(X,y_ort,M,N,C,w_ini);

                w02=W2(1);

                w12=W2(2);

                w22=W2(3);

                y2= -(w02+(w12*xax))/w22;
```

```
cd(FolderName);
hgload('2Features');
hold on;
figure(C+3)
plot(xax,y2,':k','LineWidth',2);
hold on;
%plot(xsample(2),xsample(3),'go');
hold on;
title('Discriminant Function with Perceptron Algorithm');
axis([ 1 700 1 5000])
corte2=-w02/w22;
fprintf('Corte %f',corte2)
hold on;
hgsave('WithPerceptron');
hold on
cd ..

%                 %With LMS
W3= LMS(X,M,N,C,W1,y_ort);
w03=W3(1);
w13=W3(2);
w23=W3(3);
y3= -(w03+(w13*xax))/w23;
 cd(FolderName);
hgload('2Features');
hold on
figure(C+4)
plot(xax,y3,':k','LineWidth',2);
hold on;
%plot(xsample(2),xsample(3),'go');
hold on;
title('Discriminant Function with LMS Algorithm');
axis([ 1 700 1 5000])
```

```
                corte3=-w03/w23;
                fprintf('Corte %f',corte3)

                hold on;
                hgsave('With LMS');
                hold on;
                cd ..

%
                 %With SOE
                W4=SOE(X,M,N,C) ;
                w04=W4(1);
                w14=W4(2);
                w24=W4(3);
                y4= -(w04+(w14*xax))/w24;
                cd(FolderName);
                hgload('2Features');
                figure(C+5)
                plot(xax,y4,':k','LineWidth',2);
                hold on;
                plot(xsample(2),xsample(3),'go');
                hold on;
                title('Discriminant Function with SOE Algorithm');
                axis([ 1 700 1 5000])
                corte4=-w04/w24;
                fprintf('Corte %f',corte4)
                hold on ;
                hgsave('WithSOE');
                hold on
                cd ..
```

4.Discriminant Block

```
function j =DiscriminantBlock(W1,W2,W3,W4,L,IFSVector,MaxPPDUduration)
%Discriminant Block
```

```
            for j=1:L

Thershold_Pocket(j)=W1'*[1;IFSVector(j);MaxPPDUduration(j)];

Thershold_Perceptron(j)=W2'*[1;IFSVector(j);MaxPPDUduration(j)];
                Thershold_LMS(j)=W3'*[1;IFSVector(j);MaxPPDUduration(j)];
                Thershold_SOE(j)=W4'*[1;IFSVector(j);MaxPPDUduration(j)];


            end


            fprintf('\n Number of points to be classified: %d \n',L)

                    PorcentPocket=length(find(Thershold_Pocket>0));




                Real_PorcPocket=PorcentPocket*100/L;

                PossiblesBluetooths1=100-Real_PorcPocket;

                fprintf('\n Percentual of points in the Wi-Fi Class using
the Pocket Algorithm is : %f \n',Real_PorcPocket);

                fprintf('\n Percentual of points in the Bluetooth Class
using the Pocket Algorithm is : %f \n',PossiblesBluetooths1);


                   PorcentPerceptron=length(find(Thershold_Perceptron>0));




                Real_PorcPerceptron=PorcentPerceptron*100/L;

                PossiblesBluetooths2=100-Real_PorcPerceptron;

                fprintf('\n Percentual of points in the Wi-Fi Class using
the Perceptron Algorithm is : %f \n',Real_PorcPerceptron);

                fprintf(' \n Percentual of points in the Bluetooth Class
using the Perceptron Algorithm is : %f \n',PossiblesBluetooths2);



                    PorcentLMS=length(find(Thershold_LMS>0));
```

```
                    Real_PorcLMS=PorcentLMS*100/L;

                    PossiblesBluetooths3=100-Real_PorcLMS;


                fprintf('\n Percentual of points in the Wi-Fi Class using
the LMS Algorithm is : %f \n',Real_PorcLMS);

                fprintf('\n Percentual of points in the Bluetooth Class
using the LMS Algorithm is : %f \n',PossiblesBluetooths3);


                    PorcentSOE=length(find(Thershold_SOE>0));


                    Real_PorcSOE=PorcentSOE*100/L;

                PossiblesBluetooths4=100-Real_PorcSOE;

                fprintf('\n Percentual of points in the Wi-Fi Class using
the SOE Algorithm is : %f \n',Real_PorcSOE);

                fprintf('\n Percentual of points in the Bluetooth Class
using the SOE Algorithm is : %f \n',PossiblesBluetooths4);
```

5.Error

```
%ErrorNumber
% %Multi-Slot

% W1=[27;-2951.00750966092;214];
%
% W2=[-20.9780909571695;-754754.944960720;65177.3137731307];
%
% W3=[-2.60459375983088e+40;-1.91568366881445e+40;3.80307560277537e+39];
%
% W4=[2885.28906276758;-12.2577405322002;0.213666512014766];


%SingleSlot
% W1=[-3;-1160.13813388255;614];
%
% W2=[-1124.66666666667;-596979.329122128;393714.500000000];
% W3=[-791.939264646996;-412350.058961372;241720.118250060];
% W4=[-767.507895440477;-6.77765685180857;6.59157513268478];
% %


%SingleSlot N=10000

% W1=[-11; -1951.51989846758 ; 1254];
% W2=[-729.199999999998;-296965.142763335;143747.200000000];
% W3=[-11.0557264886324;-1950.94146912034; 1160.08983006525];
% W4=[-780.131200548504; -6.91890575158495;6.67045354238516];
%
% %Multi Slot N=10000
%
% W1=[114;-7400.15315093904;619];
% W2=[438966.699999974;-894011.013611800;52211.6000000255];
% W3=[124.716346120036;-3852.40422992936;744.923713217399 ];
% W4=[1492.39002650608; -17.9499939253218;1.49834503999549];
%
```

```
C=2;

M=2;



%Wi_Fi Characterization


[TSFVectorWIFI,DurationVectorWIFI]=WIFITrainingGeneration;


[IFSVectorWIFI,MaxPPDUdurationWIFI,InterferenceWIFI]=FeatureExtraction(Durat
ionVectorWIFI,TSFVectorWIFI);


% Bluetooth Charaterization

%[DurationVectorBluetooth,TSFVectorBluetooth]=SSBluetoothTrafficGeneration;

[DurationVectorBluetooth,TSFVectorBluetooth]=MSBluetoothTrafficGeneration;



[IFSVectorBluetooth,MaxPPDUdurationBluetooth,InterferenceBluetooth]=FeatureE
xtraction(DurationVectorBluetooth,TSFVectorBluetooth);


%Calculating the Feauture's Histogram

Feature_matrix=zeros(length(IFSVectorWIFI),C*M);







              Feature_matrix(1:length(IFSVectorWIFI),1)= IFSVectorWIFI; %IFS
Wi-Fi


              Feature_matrix(1:length(IFSVectorWIFI),2)=
MaxPPDUdurationWIFI; %Max-Duration Wi-Fi


Feature_matrix(1:length(IFSVectorWIFI),3)=IFSVectorBluetooth(1:length(IFSVec
torWIFI)) ; % IFS Bluetooth


Feature_matrix(1:length(IFSVectorWIFI),4)=MaxPPDUdurationBluetooth(1:length(
IFSVectorWIFI)); %Max-Duration Bluetooth


              X1=Feature_matrix;


  [N,columna]=size(Feature_matrix) % Training Matrix
```

```
%Generation matrix of Training (M,N*C) dimension

  D=N*C;% Total Number of Training

  Xp=zeros(M,N); %Training Matrix

  X_convertion= (X1)';

  Xq=zeros(M,N*C);

  X=zeros(1:M,N*C);



  for k=1:C

       Xp(1:M,1:N)= X_convertion(1+M*(k-1):(k*M),1:N);

       Xq=[Xq Xp];

       Xp=zeros(M,N);

  end

X=Xq(1:M,(N*C)+1:2*(N*C));

Ntotal=N*C;

X=[ones(1,Ntotal); X(1:M,1:Ntotal)];

NumPocket1=zeros(1,N);

NumPocket2=zeros(1,N);

NumPerce1=zeros(1,N);

NumPerce2=zeros(1,N);

NumLMS1=zeros(1,N);

NumLMS2=zeros(1,N);

NumSOE1=zeros(1,N);

NumSOE2=zeros(1,N);

NumPocket1=zeros(1,N);

NumPocket2=zeros(1,N);


%Verification of misclassified

for i=1:N

    NumPocket1(i)= W1'*X(1:M+1,i);
    NumPerce1(i)= W2'*X(1:M+1,i);
    NumLMS1(i)= W3'*X(1:M+1,i);
    NumSOE1(i)= W4'*X(1:M+1,i);

    NumPocket2(i)= W1'*X(1:M+1,N+i);
```

```
      NumPerce2(i)= W2'*X(1:M+1,i+N);
      NumLMS2(i)= W3'*X(1:M+1,i+N);
      NumSOE2(i)= W4'*X(1:M+1,N+i);


  end

  Num11=length(find(NumPocket1>0));

  Num12=length(find(NumPocket2<0));

  Num21=length(find(NumPerce1>0));

  Num22=length(find(NumPerce2<0));

   Num31=length(find(NumLMS1>0));

  Num32=length(find(NumLMS2<0));

   Num41=length(find(NumSOE1>0));

  Num42=length(find(NumSOE2<0));


  Error11=N-Num11;

  Error12=N-Num12;

  Error21=N-Num21;

  Error22=N-Num22;

  Error31=N-Num31;

  Error32=N-Num32;

  Error41=N-Num41;

  Error42=N-Num42;

  fprintf('\n The Number of misclassified Wi-Fi second Pocket   is: %d \n
',Error11);


  fprintf('\n The Number of misclassified  Bluetooth second Pocket is: %d
\n',Error12);

   fprintf('\n The Number of misclassified Wi-Fi second Perceptron is: %d
\n',Error21);


  fprintf('\n The Number of misclassified Bluetooth second Perceptron is: %d
\n',Error22);



   fprintf('\n The Number of misclassified Wi-Fi second LMS is :%d \n
',Error31);


  fprintf('\n The Number of misclassified Bluetooth second LMS  is: %d
\n',Error32);



   fprintf('\n The Number of misclassified Wi-Fi second SOE  is: %d\n
```

```
',Error41);


  fprintf('\n The Number of misclassified Bluetooth second SOE is: %d
\n',Error42);
```

6.Feature Extraction

```
%Feature Extraction

function
[IFSVector,MaxPPDUduration,Interference]=FeatureExtraction(DurationVector,TS
FVector)


lDV = length(DurationVector)-1;

 cellIFS = 0;

 IFSVector=[];

 for DurationCounter = 1:lDV

     IFS =  TSFVector(DurationCounter+1)-TSFVector(DurationCounter)-
DurationVector(DurationCounter);

      if
(0.6*DurationVector(DurationCounter)>DurationVector(DurationCounter+1)&(IFS<
625)&(IFS>0))

          cellIFS = cellIFS+1;

          IFSVector(cellIFS)=IFS;

          CounterImportantValues(cellIFS)=DurationCounter;
      end
 end

 for fsc = 1:length(CounterImportantValues)

    if (fsc==1)

        MaxPPDUduration(1)= DurationVector(1);

        IFSPlotted(1)=IFSVector(1);
     else
        intmin = CounterImportantValues(fsc-1);

        intmax = CounterImportantValues(fsc);

        PPDUset = DurationVector(intmin:intmax);

        MaxPPDUduration(fsc) = max(PPDUset);

        IFSPlotted(fsc) = IFSVector(fsc);


    end
 end

 Interference=0;
```

7. Import Wi-Fi

```
function [TSFVector,DurationVector]=import_WiFi(filename)

%filename = '2503capture5.txt'
%PPDU duration extraction
fid = fopen (filename,'a+');
acqstring = fileread(filename);
index1 = regexp(acqstring,'Duration');
PacketsCaptured = length(index1);
index2 = regexp(acqstring,'Period                :');
indexlast = index2-1;
DurationVector = zeros(1,PacketsCaptured);

for i0 = 1:PacketsCaptured

 textdurationline = acqstring(index1(i0):indexlast(i0));
 textscannedduration = textscan(textdurationline,'%*s %*s %f %*s');
 DurationVector(i0) = cell2mat(textscannedduration);

end

%TSF extraction
fclose(fid);

fid = fopen (filename,'a+');
acqstringTSF = fileread(filename);
indexTSF1 = regexp(acqstringTSF,'TSF');
PacketsCaptured = length(indexTSF1);
indexTSF2 = regexp(acqstring,'Rate');
indexlastTSF = indexTSF2-1;
TSFVector = zeros(1,PacketsCaptured);

for j0 = 1:PacketsCaptured

 textTSFline = acqstringTSF(indexTSF1(j0):indexlastTSF(j0));
 textscannedTSF = textscan(textTSFline,'%*s %*s %f %*s');
 TSFVector(j0) = cell2mat(textscannedTSF);
end

TSFVector = TSFVector - TSFVector(1);
fclose(fid);

%Preamble extraction
fid = fopen (filename,'a+');
acqstringPre = fileread(filename);
indexPre1 = regexp(acqstringPre,'Preamble');
PacketsCaptured = length(indexPre1);
indexPre2 = regexp(acqstring,'Duration              :');
indexlastPre = indexPre2-1;
PreVector = zeros(1,PacketsCaptured);

for j0 = 1:PacketsCaptured

 textPreline = acqstringPre(indexPre1(j0):indexlastPre(j0));
 textscannedPre = textscan(textPreline, '%*s %*s %s');
 strtest = cell2str(textscannedPre);
 TF = strcmp('Long', strtest);
 if (TF == 1)
     TSFVector(j0) = TSFVector(j0)-192;

 else
     TSFVector(j0) = TSFVector(j0)-96;
```

```
 end
end

TSFVector = TSFVector - TSFVector(1);
fclose(fid);
```

8.LMS

```
% LMS Algorithm

function w= LMS(X,M,N,C,W2,y )

Ntotal=N*C;


% y=zeros(1,Ntotal);
%
% for t=1:Ntotal
%
%     y(t)=W2'*XLMS(1:(M+1),t);
%
% end


% [m,Ntotal]=size(XLMS);
%
%
% w=W2;
%
%
% rhoK= 4e-9; % Learning rate
%
% for i=1:Ntotal
%
%     w= w+((rhoK)*(y(i)-(XLMS(1:m,i))'*w)*XLMS(1:m,i));
% end

%----------------

Ntotal=N*C;

XLMSinitial=[ones(1,Ntotal); X(1:M,1:Ntotal)];

Y=[XLMSinitial(1:M+1,1:N) (-1)*XLMSinitial(1:M+1,N+1:2*N)];

w=ones(1,M+1)';

y=100*ones(Ntotal,1);

rho=1e-6;


% bmin=0.01;



for k=1:Ntotal

    error=y(k)-(w'*Y(1:M+1,k));

    errorpositive=(error+abs(error))/2;
```

```
    ynew(k)=y(k)+2*(rho)*errorpositive;

    w=w+(rho/k)*Y(1:M+1,k)*(ynew(k)-(w'*Y(1:M+1,k)));



end
```

9.Mixed Bluetooth

```
function [DurationVector,TSFVector] = MixedMSBluetoothWiFi
l=200;

[DurationVectorB,TSFVectorB]=MSBluetoothTrafficGenerationsample(l);

%[DurationVectorB,TSFVectorB]=SSBluetoothTrafficGenerationsample(l);

filename = 'VAIOHPATHSKYPEUTUBEOPENOFFICEINFOCOM.txt';

[TSFVectorW,DurationVectorW]=import_WiFi(filename);
TSFVector = sort([TSFVectorW TSFVectorB+10000]);

W=1;
B=1;
for i = 1:length(TSFVector)-1
    if (TSFVector(i)==TSFVector(i+1));
        fprintf('warning');
    end
end

for mix = 1:length(TSFVector)

    if (isempty(find(TSFVectorW==TSFVector(mix)))==0)
         DurationVector(mix)=DurationVectorW(W);
         W=W+1;
    else
        DurationVector(mix)= DurationVectorB(B);
        B = B+1;
    end
end
```

10.MS Bluetooth Traffic Generation

```
function
[DurationVectorBluetooth,TSFVectorBluetooth]=MSBluetoothTrafficGeneration

%IFs and Max PPDU Duration

    TS_duration = 625e-6;

    jitter = 10e-6;

    maxP_duration = 366e-6;

    NULL_duration = 126e-6;

    l = 20000;

    probability = 0.7;
```

```
    packet_duration = [];

    arrival_time = [];


% Scenario 3:
% 80% packets last 1 time slot
% 15% packets last 3 time slot
% 5% packets last 5 time slot

    arrival_time(1) = 0;

for i = 1:l

    if mod(i,2) == 1
        % odd packet -> data packet (master)

        chooser = rand(1,1);

        if chooser <= 0.8 % 1 time slot

            hmts = 1; % how many time slots

        elseif chooser > 0.95 % 5 time slot

            hmts = 5; % how many time slots

        else % 3 time slot

            hmts = 3; % how many time slots

        end

        switch hmts

            case 1  % 1 time slot

                if rand <= probability

                    packet_duration(i) = maxP_duration;

                else

                    packet_duration(i) = NULL_duration + 1e-
6*randint(1,1,[0,(maxP_duration-NULL_duration)*1e6]);

                end

                if i < l

                    arrival_time = [arrival_time
arrival_time(length(arrival_time))+TS_duration];

                end

            case 3  % 3 time slot

                if rand <= probability

                    packet_duration(i) = 1622e-6;

                else

                    packet_duration(i) = 2*TS_duration + 1e-
```

```
6*randint(1,1,[0,372]);

                end

                if i < l

                    arrival_time = [arrival_time
arrival_time(length(arrival_time))+3*TS_duration];

                end


            case 5  % 5 time slot

                if rand <= probability

                    packet_duration(i) = 2870e-6;

                else

                    packet_duration(i) = 4*TS_duration + 1e-
6*randint(1,1,[0,370]);

                end

                if i < l

                    arrival_time = [arrival_time
arrival_time(length(arrival_time))+5*TS_duration];

                end

        end

    else
        % even packet -> NULL packet as ACK (slave)

        packet_duration(i) = NULL_duration;

        if i < l

            arrival_time = [arrival_time
arrival_time(length(arrival_time))+TS_duration];

        end

    end


end


for i = 2:l

    j = randn*jitter/3; % 99% of values in +or- 3 sigma -> in +or- jitter

    if j < - jitter

        j = - jitter;

    end

    if j > jitter

        j = jitter;
```

```
    end

    arrival_time(i) = arrival_time(i)+j;
end

DurationVectorBluetooth = 10^6.*packet_duration;

TSFVectorBluetooth = 10^6.*arrival_time;
```

11.MS Bluetooth Traffic Generation sample

```
function
[DurationVectorBluetooth,TSFVectorBluetooth]=MSBluetoothTrafficGenerationsam
ple(l);

%IFs and Max PPDU Duration

    TS_duration = 625e-6;

    jitter = 10e-6;

    maxP_duration = 366e-6;

    NULL_duration = 126e-6;

   % l = 1000;

    probability = 0.7;

    packet_duration = [];

    arrival_time = [];


% Scenario 3:
% 80% packets last 1 time slot
% 15% packets last 3 time slot
% 5% packets last 5 time slot

    arrival_time(1) = 0;

for i = 1:l

    if mod(i,2) == 1
        % odd packet -> data packet (master)

        chooser = rand(1,1);

        if chooser <= 0.8 % 1 time slot

            hmts = 1; % how many time slots

        elseif chooser > 0.95 % 5 time slot

            hmts = 5; % how many time slots

        else % 3 time slot

            hmts = 3; % how many time slots

        end
```

```
        switch hmts
            case 1  % 1 time slot
                if rand <= probability
                    packet_duration(i) = maxP_duration;
                else
                    packet_duration(i) = NULL_duration + 1e-
6*randint(1,1,[0,(maxP_duration-NULL_duration)*1e6]);
                end
                if i < l
                    arrival_time = [arrival_time
arrival_time(length(arrival_time))+TS_duration];
                end
            case 3  % 3 time slot
                if rand <= probability
                    packet_duration(i) = 1622e-6;
                else
                    packet_duration(i) = 2*TS_duration + 1e-
6*randint(1,1,[0,372]);
                end
                if i < l
                    arrival_time = [arrival_time
arrival_time(length(arrival_time))+3*TS_duration];
                end

            case 5  % 5 time slot
                if rand <= probability
                    packet_duration(i) = 2870e-6;
                else
                    packet_duration(i) = 4*TS_duration + 1e-
6*randint(1,1,[0,370]);
                end
                if i < l
                    arrival_time = [arrival_time
arrival_time(length(arrival_time))+5*TS_duration];
                end
        end
    else
```

```
        % even packet -> NULL packet as ACK (slave)
        packet_duration(i) = NULL_duration;
        if i < l
            arrival_time = [arrival_time
arrival_time(length(arrival_time))+TS_duration];
        end
    end

end

for i = 2:l
    j = randn*jitter/3; % 99% of values in +or- 3 sigma -> in +or- jitter
    if j < - jitter
        j = - jitter;
    end
    if j > jitter
        j = jitter;
    end
    arrival_time(i) = arrival_time(i)+j;
end
DurationVectorBluetooth = 10^6.*packet_duration;
TSFVectorBluetooth = 10^6.*arrival_time;
```

12.Perce

```
% Perceptrom Algorithm
function w =perce(X,y,M,N,C,w_ini)
Ntotal=N*C;
Xper=[ones(1,Ntotal); X(1:M,1:Ntotal)];

[m,Ntotal]=size(Xper);
yper=y;
max_iter=100000; % Maximum allowable number of iterations
rho=1; % Learning rate
w=w_ini; % Initialization of the parameter vector
iter=0; % Iteration counter
mis_clas=Ntotal; % Number of misclassified vectors
```

```
while(mis_clas>0)&&(iter<max_iter)

    iter=iter+1;

    mis_clas=0;

    gradi=zeros(M+1,1) ;% Inizialitation of the gradient term

    for i=1:Ntotal

        if((Xper(:,i)'*w)*yper(i)<0) % Verification Perceptron cost

            mis_clas=mis_clas+1;

            gradi=gradi-(yper(i)*Xper(:,i)); % Computation of the gradient
term
        end

    end

    w=w-(1/iter)*gradi; %Updating the parameter vector
end

fprintf('\n Iteration Number with Perceptron %d \n',iter);
```

13.Pocket

```
% Pocket Algorithm with Ratchet

function W= Pocket(X,y,M,N,C)

%X=[] Training Feature Vectors Matrix

%y=[] Vector of desired responses

%W=[] Vector of integral pocket weights

%pi=[] Vector of integral perceptron weights

%run_pi= number of consecutive correct classifications using perceptron
weights pi

%run_w= number of consecutive correct classifications using pockets weights
W

%num_okpi= total number of training examples that pi correctly classifies.

%num_okw= total number of training examples that W correctly classifies.

Ntotal=N*C;

Xpocket=[ones(1,Ntotal); X(1:M,1:Ntotal)];


[m,Ntotal]=size(Xpocket);


pi=(zeros(1,m))';

run_pi=0;

run_w=0;

num_okpi=0;
```

```
num_okw=0;
num_Iteration=80000;
Iteration_counter=1;
index= ceil(Ntotal*rand(1,1)) ;% Randomly pick a training example
x_sample= Xpocket(1:m,index);
y_sample= y(index);


        while ((Iteration_counter<num_Iteration) && (num_okw<Ntotal))


                f_sample=pi'*x_sample;
                    if (((f_sample >0) && (y_sample ==1)) || ((f_sample <0)
&& (y_sample ==-1)))


                                run_pi=run_pi+1;
                                    if (run_pi>run_w)
                                %Compute num_okpi by checking every training
example
                                 for i=1:Ntotal
                                   f_sample_vector(i)=pi'*Xpocket(1:m,i);
                                 end
                                 thershold=f_sample_vector.*y;
                                 correctly_index=find(thershold>0);
                                 num_okpi=length(correctly_index);
                                    if (num_okpi>num_okw)
                                        W=pi;
                                        run_w=run_pi;
                                        num_okw=num_okpi;
                                            if (num_okw==Ntotal)
                                            break

                                            end
                                    end
                                    end
```

```
                        index= ceil(Ntotal*rand(1,1)); % Randomly pick a
training example
                        x_sample= Xpocket(1:m,index);
                        y_sample= y(index);
                        f_sample=pi'*x_sample;




                    else

                            pi=pi+y_sample*x_sample;
                            run_w=0;
                            run_pi=0;



                        end
            Iteration_counter=Iteration_counter+1;
        end
        fprintf('\n Iteration Number with Pocket: %d \n',Iteration_counter);
```

14.SOE

```
%Sum of Error Squares Estimation



function w= SOE(X,M,N,C)

% Ntotal=N*C;
%
% XSOE=[ones(1,Ntotal); X(1:M,1:Ntotal)];
%
%
% [m,Ntotal]=size(XSOE);
%
% y=zeros(1,Ntotal)';
%
% for t=1:Ntotal
%
%     y(t)=W2'*XSOE(1:(M+1),t);
%
% end
%
% W=(zeros(1,m))';
%
```

```
%
%
% %Compute the weights vector
%
%
%      W= inv(XSOE*XSOE')*(XSOE*y);
%

Ntotal=N*C;

XSOEinitial=[ones(1,Ntotal); X(1:M,1:Ntotal)];

Y1=XSOEinitial';

Y=[Y1(1:N,1:M+1);(-1)*Y1(N+1:2*N,1:M+1)];

w=ones(1,M+1)';

y=ones(Ntotal,1);

rho=0.9;

% bmin=0.01;

MaxIteration=1000000;


for k=1:MaxIteration

    error=(Y*w)-y;

    errorpositive=(error+abs(error))/2;

    y=y+2*(rho)*errorpositive;

    w=(inv(Y'*Y)*Y')*y;

    if(Y*w >0)

        break;

    end

end


    fprintf('K=,%d',k);
%
%     x=linspace(1,100,1000);
%
%     w1=w(1);
%
%     w2=w(2);
%
%     w3=w(3);
%
%     ynuevo =zeros(1,1000);
%
%     ynuevo =(-1)*((w2*x)+w1)/w3;
%
%     plot(x,ynuevo)
%     hold on;
%     grid on;
%
%     axis([1 50 0 20]);
```

```
%
```

15. SS Bluetooth Traffic Generation

```
                              function
[DurationVectorBluetooth,TSFVectorBluetooth]=SSBluetoothTrafficGeneration

clear
clc
close all



% Initialization

TS_duration = 625e-6;
jitter = 10e-6;
maxP_duration = 366e-6;
NULL_duration = 126e-6;

l = 20000;
probability = 0.7;

packet_duration = [];
arrival_time = [];



% Scenario 1:
% 100% packets last 1 time slot

for i = 1:l

    if mod(i,2) == 1
        % odd packet -> data packet (master)
        if rand <= probability
            packet_duration(i) = maxP_duration;
        else
            packet_duration(i) = NULL_duration + 1e-
6*randint(1,1,[0,(maxP_duration-NULL_duration)*1e6]);
        end
    else
        % even packet -> NULL packet as ACK (slave)
        packet_duration(i) = NULL_duration;
    end

    arrival_time(i) = (i-1)*TS_duration;

end

for i = 2:l
    j = randn*jitter/3; % 99% of values in +or- 3 sigma -> in +or- jitter
    if j < - jitter
        j = - jitter;
    end
    if j > jitter
        j = jitter;
    end
    arrival_time(i) = arrival_time(i)+j;
end

DurationVectorBluetooth = 10^6.*packet_duration;

TSFVectorBluetooth = 10^6.*arrival_time;
```

16.SS Bluetooth Traffic Generation sample

```
function
[DurationVectorBluetooth,TSFVectorBluetooth]=SSBluetoothTrafficGenerationsam
ple(l)

% clear
% clc
% close all
%


% Initialization

TS_duration = 625e-6;
jitter = 10e-6;
maxP_duration = 366e-6;
NULL_duration = 126e-6;

% l = 1000;
probability = 0.7;

packet_duration = [];
arrival_time = [];



% Scenario 1:
% 100% packets last 1 time slot

for i = 1:l

    if mod(i,2) == 1
        % odd packet -> data packet (master)
        if rand <= probability
            packet_duration(i) = maxP_duration;
        else
            packet_duration(i) = NULL_duration + 1e-
6*randint(1,1,[0,(maxP_duration-NULL_duration)*1e6]);
        end
    else
        % even packet -> NULL packet as ACK (slave)
        packet_duration(i) = NULL_duration;
    end

    arrival_time(i) = (i-1)*TS_duration;

end

for i = 2:l
    j = randn*jitter/3; % 99% of values in +or- 3 sigma -> in +or- jitter
    if j < - jitter
        j = - jitter;
    end
    if j > jitter
        j = jitter;
    end
    arrival_time(i) = arrival_time(i)+j;
end

DurationVectorBluetooth = 10^6.*packet_duration;

TSFVectorBluetooth = 10^6.*arrival_time;
```

17. Wi-Fi Traffic Adquisition

```
function [TSFVector,DurationVector] = WIFITrafficAcquisition(filename)

fid = fopen (filename,'a+');
acqstring = fileread(filename);
index1 = regexp(acqstring,'Duration');
PacketsCaptured = length(index1);
index2 = regexp(acqstring,'Period               :');
indexlast = index2-1;
DurationVector = zeros(1,PacketsCaptured);

for i0 = 1:PacketsCaptured

 textdurationline = acqstring(index1(i0):indexlast(i0));
 textscannedduration = textscan(textdurationline,'%*s %*s %f %*s');
 DurationVector(i0) = cell2mat(textscannedduration);

end

fclose(fid);

fid = fopen (filename,'a+');
acqstringTSF = fileread(filename);
indexTSF1 = regexp(acqstringTSF,'TSF');
PacketsCaptured = length(indexTSF1);
indexTSF2 = regexp(acqstring,'Rate');
indexlastTSF = indexTSF2-1;
TSFVector = zeros(1,PacketsCaptured);

for j0 = 1:PacketsCaptured

 textTSFline = acqstringTSF(indexTSF1(j0):indexlastTSF(j0));
 textscannedTSF = textscan(textTSFline,'%*s %*s %f %*s');
 TSFVector(j0) = cell2mat(textscannedTSF);
end

TSFVector = TSFVector - TSFVector(1);
fclose(fid);
```

18.Wi-Fi Training Generation

```
function[TSFVectorWIFI,DurationVectorWIFI]=WIFITrainingGeneration


filename1 = 'VAIOHPATHSKYPEUTUBEOPENOFFICEINFOCOM3.txt';

[TSFVector1,DurationVector1]=import_WiFi(filename1);

filename2 = 'VAIOHPATHSKYPEUTUBEOPENOFFICEINFOCOM2.txt';

[TSFVector2,DurationVector2]=import_WiFi(filename2);


filename3 = 'VAIOHPdv6000DownINFOCOM.txt';

[TSFVector3,DurationVector3]=import_WiFi(filename3);


filename4 = 'VAIODownINFOCOM.txt';
```

```
[TSFVector4,DurationVector4]=import_WiFi(filename4);

filename5 = 'VAIOATHSKYPEUTUBEINFOCOM3.txt';

[TSFVector5,DurationVector5]=import_WiFi(filename5);

filename6 = 'VAIOATHSKYPEUTUBEINFOCOM2.txt';

[TSFVector6,DurationVector6]=import_WiFi(filename6);

filename7 = '2503capture1.txt';

[TSFVector7,DurationVector7]=import_WiFi(filename7);

filename8 = '2503capture2.txt';

[TSFVector8,DurationVector8]=import_WiFi(filename8);

filename9 = '2503capture3.txt';

[TSFVector9,DurationVector9]=import_WiFi(filename9);

filename10 = '2503capture4.txt';

[TSFVector10,DurationVector10]=import_WiFi(filename10);

filename11 = '2503capture5.txt';

[TSFVector11,DurationVector11]=import_WiFi(filename11);

filename12 = '2503capture6.txt';

[TSFVector12,DurationVector12]=import_WiFi(filename12);

filename13 = '2503capture7.txt';

[TSFVector13,DurationVector13]=import_WiFi(filename13);

filename14 = '2503capture8.txt';

[TSFVector14,DurationVector14]=import_WiFi(filename14);

filename15 = '2503capture9.txt';

[TSFVector15,DurationVector15]=import_WiFi(filename15);

filename16 = '2503capture10.txt';

[TSFVector16,DurationVector16]=import_WiFi(filename16);


DurationVectorWIFI=[DurationVector1 DurationVector2 DurationVector3
DurationVector4 DurationVector5 DurationVector6 DurationVector7
DurationVector8 DurationVector9 DurationVector10 DurationVector11
```

```
DurationVector12 DurationVector13 DurationVector14  DurationVector16];

TSFVector2 =
TSFVector2+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVec
tor1))+ 3000;

TSFVector3 =
TSFVector3+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVec
tor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector2
))+3000;
TSFVector4 =
TSFVector4+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVec
tor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector2
))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+3
000;
TSFVector5 =
TSFVector5+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVec
tor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector2
))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+T
SFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+3000;
TSFVector6 =
TSFVector6+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVec
tor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector2
))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+T
SFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFVe
ctor5(length(TSFVector5))+DurationVector5(length(DurationVector5))+3000;

TSFVector7 =
TSFVector7+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVec
tor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector2
))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+T
SFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFVe
ctor5(length(TSFVector5))+DurationVector5(length(DurationVector5))+TSFVector
6(length(TSFVector6))+DurationVector6(length(DurationVector6))+3000;
TSFVector8=
TSFVector8+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVec
tor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector2
))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+T
SFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFVe
ctor5(length(TSFVector5))+DurationVector5(length(DurationVector5))+TSFVector
6(length(TSFVector6))+DurationVector6(length(DurationVector6))+
TSFVector7(length(TSFVector7))+DurationVector7(length(DurationVector7))+3000
;

TSFVector9=
TSFVector9+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVec
tor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector2
))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+T
SFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFVe
ctor5(length(TSFVector5))+DurationVector5(length(DurationVector5))+TSFVector
6(length(TSFVector6))+DurationVector6(length(DurationVector6))+
TSFVector7(length(TSFVector7))+DurationVector7(length(DurationVector7))+TSFV
ector8(length(TSFVector8))+DurationVector8(length(DurationVector8))+3000;

TSFVector10=
TSFVector10+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVe
ctor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector
2))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+
TSFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFV
ector5(length(TSFVector5))+DurationVector5(length(DurationVector5))+TSFVecto
r6(length(TSFVector6))+DurationVector6(length(DurationVector6))+
TSFVector7(length(TSFVector7))+DurationVector7(length(DurationVector7))+TSFV
ector8(length(TSFVector8))+DurationVector8(length(DurationVector8))+TSFVecto
r9(length(TSFVector9))+DurationVector9(length(DurationVector9))+3000;

TSFVector11=
```

```
TSFVector11+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVe
ctor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector
2))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+
TSFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFV
ector5(length(TSFVector5))+DurationVector5(length(DurationVector5))+TSFVecto
r6(length(TSFVector6))+DurationVector6(length(DurationVector6))+
TSFVector7(length(TSFVector7))+DurationVector7(length(DurationVector7))+TSFV
ector8(length(TSFVector8))+DurationVector8(length(DurationVector8))+TSFVecto
r9(length(TSFVector9))+DurationVector9(length(DurationVector9))+
TSFVector10(length(TSFVector10))+DurationVector10(length(DurationVector10))+
3000;
TSFVector12=
TSFVector12+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVe
ctor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector
2))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+
TSFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFV
ector5(length(TSFVector5))+DurationVector5(length(DurationVector5))+TSFVecto
r6(length(TSFVector6))+DurationVector6(length(DurationVector6))+
TSFVector7(length(TSFVector7))+DurationVector7(length(DurationVector7))+TSFV
ector8(length(TSFVector8))+DurationVector8(length(DurationVector8))+TSFVecto
r9(length(TSFVector9))+DurationVector9(length(DurationVector9))+
TSFVector10(length(TSFVector10))+DurationVector10(length(DurationVector10))+
TSFVector11(length(TSFVector11))+DurationVector11(length(DurationVector11))+
3000;
TSFVector13=
TSFVector13+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVe
ctor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector
2))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+
TSFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFV
ector5(length(TSFVector5))+DurationVector5(length(DurationVector5))+TSFVecto
r6(length(TSFVector6))+DurationVector6(length(DurationVector6))+
TSFVector7(length(TSFVector7))+DurationVector7(length(DurationVector7))+TSFV
ector8(length(TSFVector8))+DurationVector8(length(DurationVector8))+TSFVecto
r9(length(TSFVector9))+DurationVector9(length(DurationVector9))+
TSFVector10(length(TSFVector10))+DurationVector10(length(DurationVector10))+
TSFVector11(length(TSFVector11))+DurationVector11(length(DurationVector11))+
TSFVector12(length(TSFVector12))+DurationVector12(length(DurationVector12))+
3000;
TSFVector14=
TSFVector14+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVe
ctor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector
2))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+
TSFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFV
ector5(length(TSFVector5))+DurationVector5(length(DurationVector5))+TSFVecto
r6(length(TSFVector6))+DurationVector6(length(DurationVector6))+
TSFVector7(length(TSFVector7))+DurationVector7(length(DurationVector7))+TSFV
ector8(length(TSFVector8))+DurationVector8(length(DurationVector8))+TSFVecto
r9(length(TSFVector9))+DurationVector9(length(DurationVector9))+
TSFVector10(length(TSFVector10))+DurationVector10(length(DurationVector10))+
TSFVector11(length(TSFVector11))+DurationVector11(length(DurationVector11))+
TSFVector12(length(TSFVector12))+DurationVector12(length(DurationVector12))+
TSFVector13(length(TSFVector13))+DurationVector13(length(DurationVector13))+
3000;

 TSFVector15=
TSFVector15+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVe
ctor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector
2))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+
TSFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFV
ector5(length(TSFVector5))+DurationVector5(length(DurationVector5))+TSFVecto
r6(length(TSFVector6))+DurationVector6(length(DurationVector6))+
TSFVector7(length(TSFVector7))+DurationVector7(length(DurationVector7))+TSFV
ector8(length(TSFVector8))+DurationVector8(length(DurationVector8))+TSFVecto
r9(length(TSFVector9))+DurationVector9(length(DurationVector9))+
TSFVector10(length(TSFVector10))+DurationVector10(length(DurationVector10))+
TSFVector11(length(TSFVector11))+DurationVector11(length(DurationVector11))+
```

```
TSFVector12(length(TSFVector12))+DurationVector12(length(DurationVector12))+
TSFVector13(length(TSFVector13))+DurationVector13(length(DurationVector13))+
TSFVector14(length(TSFVector14))+DurationVector14(length(DurationVector14))+
3000;

TSFVector16=
TSFVector16+TSFVector1(length(TSFVector1))+DurationVector1(length(DurationVe
ctor1))+TSFVector2(length(TSFVector2))+DurationVector2(length(DurationVector
2))+TSFVector3(length(TSFVector3))+DurationVector3(length(DurationVector3))+
TSFVector4(length(TSFVector4))+DurationVector4(length(DurationVector4))+TSFV
ector5(length(TSFVector5))+DurationVector5(length(DurationVector5))+TSFVecto
r6(length(TSFVector6))+DurationVector6(length(DurationVector6))+
TSFVector7(length(TSFVector7))+DurationVector7(length(DurationVector7))+TSFV
ector8(length(TSFVector8))+DurationVector8(length(DurationVector8))+TSFVecto
r9(length(TSFVector9))+DurationVector9(length(DurationVector9))+
TSFVector10(length(TSFVector10))+DurationVector10(length(DurationVector10))+
TSFVector11(length(TSFVector11))+DurationVector11(length(DurationVector11))+
TSFVector12(length(TSFVector12))+DurationVector12(length(DurationVector12))+
TSFVector13(length(TSFVector13))+DurationVector13(length(DurationVector13))+
TSFVector14(length(TSFVector14))+DurationVector14(length(DurationVector14))+
TSFVector15(length(TSFVector15))+DurationVector15(length(DurationVector15))+
3000;
TSFVectorWIFI=[TSFVector1 TSFVector2 TSFVector3 TSFVector4 TSFVector5
TSFVector6 TSFVector7 TSFVector8 TSFVector9 TSFVector10 TSFVector11
TSFVector12 TSFVector13 TSFVector14  TSFVector16 ];
```