INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

A sound based data-link protocol for network control evasion on Android

TESIS

Que para obtener el grado de:
Maestría en Ciencias de la Computación

PRESENTA:
Ing. Miguel Daniel Reyes Martínez

Directores de tesis:
Dr. Eleazar Aguirre Anaya
Dr. Ponciano Jorge Escamilla Ambrosio

Diciembre 2015

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:00 horas del día 04 del mes de diciembre de 2015 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"A sound based data-link protocol for network control evasion on Android"

Presentada por el alumno:

REYES	MARTÍNEZ	MIGUEL DANIEL
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	3	0	1	2	3

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

[illegible]
Dr. Eleazar Aguirre Anaya

[illegible]
Dr. Ponciano Jorge Escamilla Ambrosio

[illegible]
Dr. Raúl Acosta Bermejo

[illegible]
Dr. Rolando Menchaca Méndez

[illegible]
Dr. Moisés Salinas Rosales

Sergio Sandoval
M. en C. Sergio Sandoval Reyes

ESTADOS UNIDOS MEXICANOS

PRESIDENTE DEL COLEGIO DE PROFESORES

INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
DIRECCIÓN

[illegible]
Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 10 del mes de diciembre del año 2015, el (la) que suscribe Miguel Daniel Reyes Martínez alumno (a) del Programa de Maestría en Ciencias de la Computación con número de registro B130123, adscrito a Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Eleazar Aguirre Anaya y Dr. Ponciano Jorge Escamilla Ambrosio y cede los derechos del trabajo intitulado "A sound based data-link protocol for network control evasion on Android", al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección b130123@sagitario.cic.ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Daniel

Miguel Daniel Reyes Martínez

Nombre y firma

Resumen

Con la proliferación de los dispositivos móviles estos se han convertido en una parte integral de la vida cotidiana abriendo nuevas posibilidades de comunicación. Este tipo de dispositivos incluyen diversos subsistemas como sensores y actuadores que pueden ser usados para diferentes propósitos, por ejemplo la adquisición de datos del ambiente para ofrecer una mejor experiencia de usuario. En la practica estos subsistemas pueden ser utilizados para transmitir información mediante canales alternativos de comunicación que carecen de controles de red. En este trabajo se proponen dos protocolos tanto de capa física como de enlace de datos que en conjunto se denominan SoundComm-CISEG, estos protocolos utilizan el micrófono y los altavoces incluidos en los dispositivos móviles Android para establecer comunicación además de estar basados en el estándar IEEE 802. Estos protocolos son capaces de evadir controles de red evitando la transmisión por radiofrecuencia.

El diseño del protocolo se realizo mediante una metodología experimental, todas las especificaciones del protocolo fueron validadas mediante experimentación y teniendo en mente las limitaciones de hardware en los dispositivos móviles Android. El protocolo alcanza tasas de transferencia mas altas que el estado del arte y el diseño permite la operación del mismo en presencia de ruido ambiental. Todas estas características muestran que el protocolo puede ser utilizado en ambientes donde evadir la infraestructura tradicional de red así como sus controles de seguridad es imperativo.

Abstract

With the proliferation of mobile devices they have become an integral part of everyday life, enabling new possibilities of communication. This kind of devices may include a plethora of subsystems such as sensors and actuators which can be used for diverse purposes, for example, the acquisition of environmental data for context-aware applications. In practice, these subsystems can be used to transmit data in alternative channels with the absence of network controls. In this work, a physical layer and data-link protocols which use the built-in microphone and speakers to establish communication is presented, these protocols are based on the IEEE 802 standard and the proposed stack is called SoundComm-CISEG. This channel is capable of out-of-band transmission evading standard network controls between mobile devices.

The design of the protocol was done taking as base an experimental methodology, all specifications of the protocol were validated through tests and having in mind the resource constraints of the Android platforms. The protocol is capable to achieve higher bitrates than the state of the art and the design allows its operation in the presence of environmental noise. All these characteristics show that the protocol could be used to transmit data using an acoustic carrier in many scenarios where avoiding traditional network infrastructure and security mechanisms is imperative.

Acknowledgements

We want to acknowledge to IPN, CIC and CONACyT for the support received on this research, the supervisors and all people involved in one way or another in the realization of this work.

Contents

List of Figures

List of Tables

Introduction

In the present work a data-link protocol SoundComm-CISEG is proposed, it uses sound as a carrier signal and is an alternative way to establish communication avoiding the standard network controls, this can be useful on certain situations where the users want to avoid traditional radio frequency security mechanisms. The Android devices often include sensors and actuators which are used by the applications to provide different functionalities (context-aware applications for example), these devices are not often treated as communication devices. In this work, we use the speakers and the microphone included on Android mobile devices to establish communication between devices in order to evade network controls. There are some works exploring this idea, but at the moment a proposal of a protocol is inexistent. The protocol in this work was designed using an experimental methodology and taking as base the IEEE 802 standard as an architectural reference.

The structure of the work is the following, first, the state of the art is presented with some similar proposals, then the general structure of the protocol and their functionalities are described, these specifications and functionalities were validated experimentally with a testbed, the details of this implementation can be seen on chapters 4 and 5. Finally, various tests were performed showing the behavior of the protocol on different conditions, these tests aims to show the capacity of adaptation of the proposal to different scenarios. At the end a comparison with the state of the art is shown and the main differences explained.

Chapter 1

Communication protocols and reference models

The communication systems are complex systems, each element of them serves a specific function and interacts with other elements making them huge systems in terms of functionalities, so the implementation is not trivial. In order to overcome these problems, some models which abstract the operation of these systems are used in the form of services, layered architectures and reference models. On this chapter, an introduction to these abstractions and models is given.

1.1 Communication protocol characteristics

A communication protocol is a behavior convention that defines the temporal order of the interactions between the peer entities as well as the format (syntax and semantics) of the messages exchanged [4]. The communication protocols have diverse elements on which they rely on, these elements define how the parts involved in communication interact and the format of the interaction.

The process of exchange data between two systems can be treated as a service, the model of these services can be seen in figure 1.1, a communication service is comprised by the following parts:

- **Service users:** The communication service is requested by this part.

- **Service provider:** It offers the service to the service user.

- **Service access points:** A service user invokes a service to the service provider via service access points (SAPs).
- **Service primitives:** They are abstractions for describing interactions between the service users and service providers at the service interface. These abstractions are independent of the platform.

The mentioned services are provided by entities, which are active objects who perform the communication exchanging messages, the interaction between these entities provide the service, as can be seen in figure 1.2.

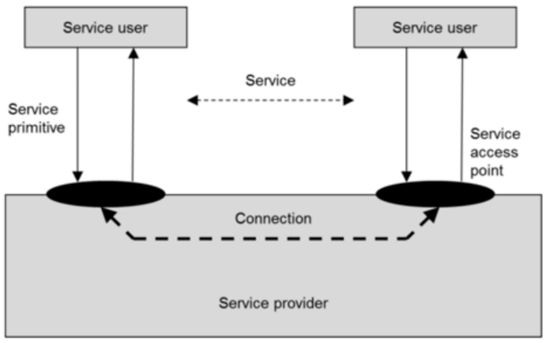


Figure 1.1: Communication services model [1]

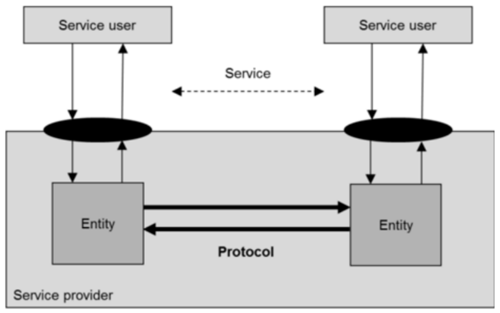


Figure 1.2: Entities interaction to provide a service [1]

1.1.1 Layered Architectures

Today, modern communication systems are constructed following the concept of layering. A layer is comprised by entities which provide the same functionality. These layers provide one or more services. On this kind of architecture, all layers provide services to the immediately adjacent higher layer, it is, if certain layer is defined as the N layer, this N-layer provides one or more services to the adjacent higher layer, the N+1 layer, in a similar fashion, the N-1 layer provides one or more services to the N layer. A protocol always defines the communication rules of entities of the same layer level as in figure 1.2, these entities are known as peer entities. The main advantage of a layered architecture is the principle of transparency, it is that the processes in each layer are transparent to the user, for example if the user data is given to the service provider, the service provider manipulates the data according to the primitives issued by the service user, the service provider is unaware of the data content, it just delivers the data unchanged to the receiver so the process is transparent to the service user.

The messages exchanged by peer entities are known as Protocol Data Units, they have an specific format in order to interpret them in the same way on both sides, a protocol usually uses at least one PDU for data transfer, other types of PDU are used for various procedures, e.g., connection establishment. The protocols are defined depending on how each PDU is processed in relationship with the later, if each PDU is treated independently, the protocol is connectionless, in the opposite case, the protocol is connection-oriented, e.g., the last case is used when the protocol has dynamic parameters regulated through the duration of the communication. Other important characteristic of the protocols is how they operate depending on the behavior of the receiver and transmitter, in this aspect the communications protocols are subdivided on two categories, symmetric and asymmetric.

- **Symmetric:** A protocol is symmetric if both communicating entities show the same behavior, e.g., in a duplex communication when both entities exchange data on both directions.

- **Asymmetric:** The protocol is asymmetric if both communicating entities show different behavior, e.g., on a simplex communication which implies communication in one direction.

1.2 Reference Models of layered architectures

The reference models describe the components of the communication architecture and the interaction principles applied. They further define the number of layers, their functionality and often, they specify the protocols used on each layer [5].

Some examples of common reference models are the OSI reference model and the model used as base in this work, the IEEE 802, this model will be explained in the section 1.3.

1.3 The IEEE 802 reference model

The IEEE 802 reference model is based on the mentioned OSI model, this model is centered on the two lower layers of the OSI model, the physical layer and the data link layer, a diagram showing the equivalence of these two reference models can be seen in figure 1.3.

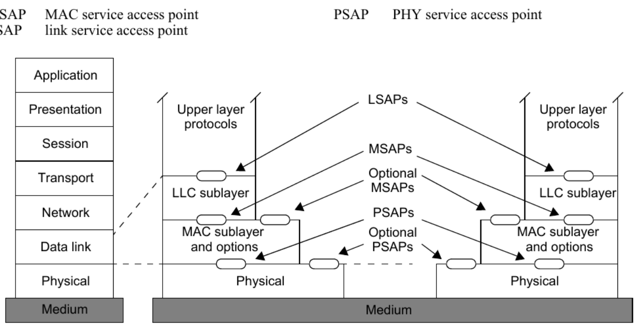


Figure 1.3: IEEE 802 reference model and it's equivalence with the OSI model[2]

1.3.1 Logical Link Control (LLC)

The LLC is a sublayer of the data-link layer which manages the logic of the communication and offers various services, the structure of this sublayer is in the figure 1.4.

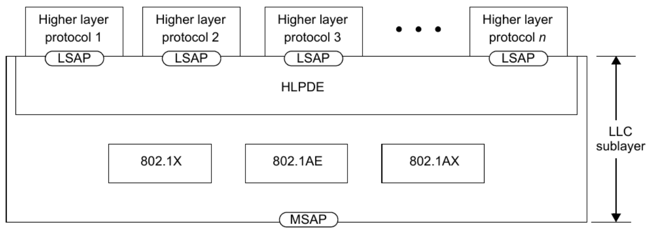


Figure 1.4: Logical Link Control [2]

The first element of the LLC is known as the Higher Layer Protocol Discrimination entity (HLPDE), this entity discriminates protocols at network layer with the purpose of delivering the data at the correct LSAP. In general terms, the LLC is concerned in the multiplexation and demultiplexation of network protocols, and also the LLC determines the flow control according to type of service of operation, they are listed next [6].

- **Unacknowledged connectionless-mode services:** This set of data transfer services provides the means by which network entities can exchange data without the establishment of a data link level connection. The data transfer can be point-to-point, multicast, or broadcast.

- **Connection-mode services:** This set of services provides the means for establishing, using, resetting, and terminating data link layer connections. These connections are point-to-point connections between LLC SAPs.

  - The connection establishment service provides the means by which a network entity can request, or be notified of, the establishment of data link layer connections.
  - The connection-oriented data transfer service provides the means by which a network entity can send or receive data over a data link layer connection. This service also provides data link layer sequencing, flow control, and error recovery.
  - The connection reset service provides the means by which established connections can be returned to the initial state.
  - The connection termination service provides the means by which a network entity can request, or be notified of, the termination of data link layer connections.
  - The connection flow control service provides the means to control the flow of data associated with a specified connection, across the network layer/data link layer interface.

- **Acknowledged connectionless-mode services:** The acknowledged connectionless-mode data unit exchange services provide the means by which network layer entities can exchange data that are acknowledged at the LLC sublayer, without the establishment of a data link connection. The services provide a means by which a network layer entity at one station can send a data unit to another station, request a previously prepared data unit from another station, or exchange data units with another station. The data unit transfer is point-to-point.

1.3.2 MAC sublayer

The MAC sublayer performs the functions necessary to provide frame-based, connectionless-mode (datagram style) data transfer between stations in support of the next higher sublayer[2].

This sublayer provides the following functions:

- Frame delimiting and recognition
- Addressing of destination stations (both as individual stations and as groups of stations)
- Conveyance of source-station addressing information
- Transparent data transfer of PDUs from the next higher sublayer
- Protection against errors, generally by means of generating and checking frame check sequences
- Control of access to the physical transmission medium

1.3.3 Physical layer

MAC entities use their respective physical layer entities to exchange bits with their peers. The physical layer entity provides the capability to transmit and receive modulated signals assigned to specific frequency channels for broadband or wireless media or to a single baseband-channel [2]. In the IEEE 802.11 [7] standard two sublayers are defined at physical layer:

- **Physical Medium Dependent sublayer (PMD):** Defines the characteristics of, and method of transmitting and receiving data through the physical medium via signals between two or more entities.
- **Physical layer convergence procedure sublayer (PLCP):** Provides a convergence mechanism, which adapts the capabilities of the PMD system to offer the physical layer services. It defines a method of mapping the IEEE 802.11 MAC PDUs into a framing format suitable for sending and receiving user data and management information between two or more entities using the associated PMD system.

Other functions like interconnection and networking are also specified, but they are not relevant on our present work since the protocol presented here is assumed point to point, other functions and parameters which do not apply to our protocol are explained on next chapters.

On this chapter the concepts and terminology of the communication protocols were introduced and the IEEE reference model as well, all these concepts are used on the research. In the present work due the nature of the proposed protocol the data-link layer operates in unacknowledged connectionless-mode, this is due the fact that the proposal is a simplex and point to point protocol, other modes of operation are impossible considering the mentioned assumptions. These limitations do not interfere at the physical layer which is considered fully functional in this work. In the next chapter we present similar approaches to the present work and a brief comparison is given.

Chapter 2

Alternative communication channels for the evasion of controls on Android mobile devices

Mobile devices are an important part of the common technological ecosystem, these kind of devices are sold on greater numbers than the PCs [8], and are an ubiquitous part of the life of people in many ways. These mobile platforms are transforming the way we interact with our environment, for example IoT, mobile banking, mobile payments and applications in general which are focused on the broad market that mobile devices have constructed today.

Mobile devices have particular characteristics that are not common on the traditional PC paradigm, for example they are equipped with a wide range of sensors, these sensors are used for data acquisition, geolocalization and other purposes limited by the creativity and the physical constraints of them. These devices have also actuators capable of interact with the environment, for example speakers and vibrators.

This capacity of receiving and sending data across an alternative channels, makes easier to evade or even avoid the traditional control mechanisms by simply using another channel which lacks of the proper security controls. The possibility of exploitation of these alternative channels is empowered with the plethora of sensors and actuators built on mobile devices, a characteristic often missed in traditional PCs. Many alternative channels are constructed over the use of subsystems included in mobile devices, i.e. the present work which makes use of microphones and speaker to establish communication and effectively evade standard network controls.

2.1 Alternative channels for local communication

On Marforio *et al.* [9] several alternative channels for local communication are presented for Android platforms, the first one uses the classical method for communication between applications on Android OS, the broadcast intents, these kind of intents are used to transmit information to a particular application using the application ID, on this research, the values that the broadcast intent hold, could be used to encode information in a covert way, making possible to transmit secret data to another application.

Another method, is to use the free space on the file system as a medium for data encoding. The transmitter writes information until certain threshold, the receiver reads how much free space remains on the file system, different values are used to encode a 1 and a 0. In a similar fashion of encoding, the CPU frequency can be modified via intensive computations in order to communicate information between two applications, one changing the CPU frequency and another monitoring the change.

Alternative channels for local communication can also be established with sensors. In Al-Haiqi *et al.* [10] the vibration is used to cause changes on the accelerometer. The transmitter uses certain patterns of vibration that can be sensed on the receiver side via sensor events, both applications could easily have permissions to perform the task, and additional considerations should be taken to remain unnoticeable to users, for example, trigger the communication at night and/or while the user is away.

In Lalande *et al.* [11] four alternative channels for local communication on Android are proposed. The first channel uses the time elapsed since the screen is switched off, which is a state that can be detected by the applications using the GET_TASK permission. The transmitter (A) waits the screen to turn off $x * \Delta T$ seconds where x is the encoded information while ΔT is a constant known by the transmitter and the receiver (B). B is capable of check the status of A. When $x * \Delta T$ seconds have passed, the process A kills itself, B detects that A is no longer running and is capable of infer x from the point where the screen went off to the moment when A stopped.

A second variant of the mentioned channel is to change the process priority p of the process A instead of kill the process A, this procedure has the advantage of the lack of GET_TASK permission. The third variant is pure process priority based. The receiver just waits for the appearance of certain priority p provided by A, and then starts to count the time $x * \Delta T$ to infere x.

The fourth solution is to use only the state of the screen, the receiver just counts the time from the point that it went off, to the point that is turned on by the transmitter.

2.2 Alternative channels with external communication

On Fernandes *et al.* [12], an scenario of a channel with external interaction is presented, an Android device equipped with an FM receiver downloads a malicious application from the market, this application itself does not contain malicious code in order to avoid security controls, once placed on the phone, the application starts to sniff information from the FM receiver, the attackers with special equipment for FM transmission, start to transmit the malicious payload, in this way several security controls are evaded starting from those which perform analysis at installation time and those who regulate network communication as well.

2.3 Sound based communication on Android Devices

There are some communication channels which make use of sound to transmit data, these covert channels use the microphone and speakers as receiver and transmitter respectively. Similar works are mentioned in order to establish a context and contribution of our work.

On Hasan *et al.* [13], control commands are propagated via external environmental changes (light, sound, vibration, magnetism), such changes are sensed on the mobile device who hosts the receiver, and it acts in response. These environmental stimulus were produced by different devices which can be used to transmit signals, for example, the light of a TV, a PC screen, PC speakers, etc. In the case of sound, distances of 55 feet indoors and 45 feet outdoors were obtained with the sound propagated command using a 17 KHz carrier.

Other covert channels which make use of sound as a information signal is presented by Deshotels [14]. In this research, a proof of concept of feasibility is shown, a sound file FSK modulated is created on a PC via MATLAB software, then, this file is played on one device and received on the other, this paper only addresses the problem of feasibility. A small resume of these proposals is in table 2.1.

Proposal	**Characteristics**
Sensing-enabled Channels for Hard-to-detect Command and Control of Mobile Devices [13]	• Transmission with PC speakers • Communication from a PC to a Mobile device. • Bitrate: 1 bit/second • Hardware – HTC Evo 4g – Android 2.3.3 – Qualcomm MSM8655 1.2 GHz – 768 MB RAM • 17 KHz carrier signal
Inaudible Sound as a Covert Channel in Mobile Devices [14]	• Transmission from mobile device to mobile device. • Communication using wav files created with MATLAB. • Maximum bitrate: 345 bit/second • Hardware – Nexus 7 tablet – Android 4.3 – Qualcomm Snapdragon S4 Pro 8064 Quad-Core, 1.5 GHz – 2 GB RAM • 18khz and 19khz frequencies

Table 2.1: Similar proposals

In this work we present a data-link and physical layer protocol which are based on IEEE 802 reference model, the main difference with the mentioned works in this section is that we define the operation of a real inter-device communication protocol with a sound carrier on an implemented testbed, in contrast with the state of the art which only shows proofs of concept to prove feasibility. In the next chapter we present the structure and the operation of this sound based protocol called SoundComm-CISEG.

Chapter 3

SoundComm-CISEG

The possibilities of environmental data acquisition on Android devices enable the capacity of receiving information through the sensor devices and the collected information can be sent to somewhere outside the mobile device through the built-in actuators (for example sensing light levels and playing a sound, respectively). These new characteristics make possible the communication on both directions, from an external entity to the mobile device and from the mobile device to an external entity. This kind of mechanism may be useful to transmit sensible data in a covert manner.

The advantage of sending sensible data in a sound based channel is the absence of proper security controls on the channel. This is, there is no way to detect the transmission with the usual network infrastructure, it could be even more complicated to try to detect it whether the data is transmitted on high frequencies, which are less notorious to human ear. However, with the advantages, there are disadvantages, for example, the distance of transmission: both transmitter and receiver cannot be geographically distant due the physical limitations of sound propagation. Despite the limitations, this kind of characteristics may be very useful on specific scenarios, for example acquiring sensible data for industrial espionage on a targeted attack during a meeting, or evasion of highly secured (or insecure) communication channels for data sharing.

As we stated in chapter 2, some alternative channels were designed to use sensors as a tool for data reception, in the present work sensors and actuators are used to establish communication, in this case, the speaker of the android device for transmission, and the microphone for reception. There is a difference to notice since in previous works only proofs of concept were presented with this idea, in this work a completely functional data-link protocol called SoundComm-CISEG was designed. In the next sections all the characteristics of operation and design are explained.

3.1 Scenario of application

Due to the physical limitations of sound propagation and hardware capacity, the distance that can be reached with this method is limited. Despite this limitation there are some scenarios where this kind of channel can be applied, specifically those associated with transmission in short distances. Consider for example an indoors scenario where the environmental sound levels and frequencies are not too high to interfere with the sound signal, for example a meeting room. Under these conditions a smartphone could initiate communication with another device using sound. The sound volume will not be as high to be noticed by the assistants, and additionally, a high frequency sound will be less perceptible to the human ear. The environmental sound and the assistant's speech will help to keep the communication between devices masked and as stealthy as possible. Under these circumstances the communication can be established in a covert manner for diverse purposes. A communication on the stated environment could be used for diverse purposes, for example, in a malware scenario of data leakage, a malware with sound communication capabilities may be listening for a signal from another device to start sending information through the channel as fast as possible in order to avoid being noticed by people near the communication location. Similar characteristics apply to a data sharing scenario, the main difference lies in the fact that both users are aware of the communication taking place, on these conditions the stealthiness could be irrelevant. In both scenarios several network controls are avoided just not using common network infrastructures. Transmitting via sound could be useful in order to maintain the communication unmonitored.

3.2 Design requirements

As we don't want to adapt special hardware in order to achieve communication, the design requirements are shaped by hardware limitations. Additional considerations should be taken for the scenarios of application, the most important ones are those related with the stealthiness, as much of the functionality of the covert channel lies in the assumption of the communication being not detected.

Two important points about the stealthiness of the channel are the frequency that will be used to transmit the data, and the time needed to send the information, we want to transmit as fast as possible because we want to minimize the possibility of being heard and being noticed during data transmission. The frequency of transmission should be carefully considered due to the implications of the physical system, for example, the maximum distance between transmitter and receiver on which the communication is established will change depending on the operation frequency.

The last point to consider is the organization of the protocol. In order to achieve communication, modularity and also extensibility, certain functionalities are necessary to be addressed, the architecture and functionalities are provided by the existing reference models, for example the IEEE 802 [2] which is a largely used standard for data-link communications.

The characteristics to be considered are:

- Channel capacity depends on the hardware limitations of mobile devices.
- The transmission times should be short in order to maintain the communication as stealthy as possible.
- The frequency of sound used for transmission should be as high as possible in order to make the transmission more stealthy (higher frequencies are less perceptible to the human ear).
- The need to address channel interferences, for example environmental noise, in order to make the protocol more resilient.
- The IEEE 802 standard is used as reference for SoundComm-CISEG which is a simplex and point to point stack of protocols.

All mentioned specifications will give shape to the subsequent characteristics of the protocol.

3.3 Research methodology

The design process is based on an experimental methodology, since we wanted to adapt all protocol features to the existing Android hardware. The research process followed is illustrated in figure 3.1, the first step was the analysis of the existing solutions which perform communication using a sound carrier involving Android devices. The second step was the analysis of the existing tools to manipulate the hardware in order to construct the testbed which is used to validate the protocol. The third step is the analysis of the possible modulation methods at the physical layer since there are various methods involving different processes at the demodulator, the simplest method was chosen having in mind the resource constraints of the platform. Having chosen the method of modulation, the process of design starts, it is represented as a loop of design, implementation and test at the physical layer and data-link layer as well. As we mentioned earlier, due to the practical nature of the present work, several approaches were proposed and the best was selected, the same process was followed for both layers. At the end some functionality test were done in order to acknowledge certain characteristics of the protocol.

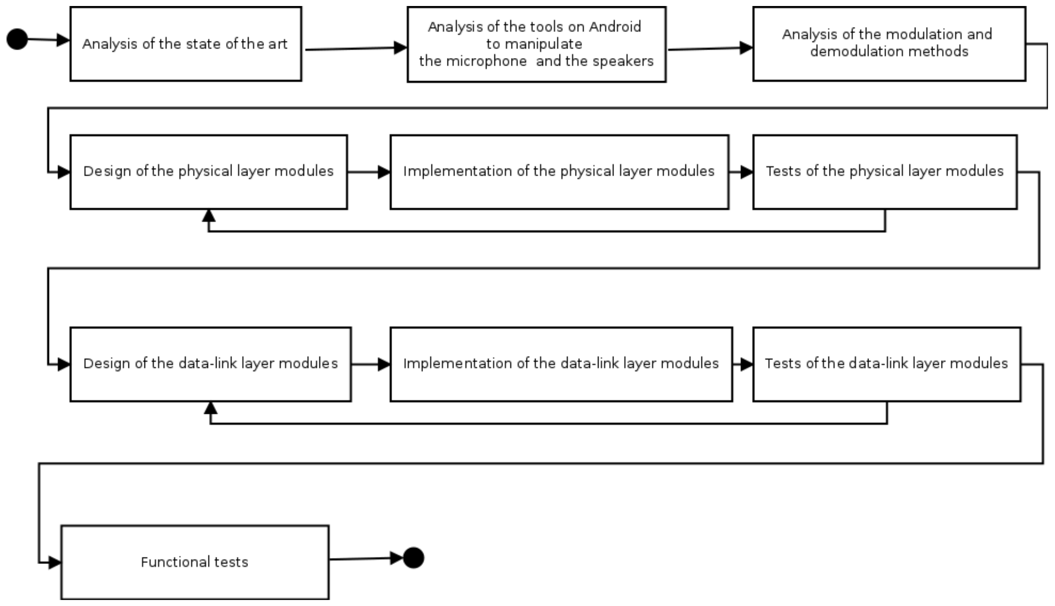

Analysis of the state of the art
Analysis of the tools on Android to manipulate the microphone and the speakers
Analysis of the modulation and demodulation methods
Design of the physical layer modules
Implementation of the physical layer modules
Tests of the physical layer modules
Design of the data-link layer modules
Implementation of the data-link layer modules
Tests of the data-link layer modules
Functional tests

Figure 3.1: Design methodology.

3.4 Design scenario

The design process involves the validation of certain characteristics in a practical way. In order to validate the specifications, an testbed was developed, this testbed makes use of APIs who control the hardware (in this case, the microphone and the speakers). The details of the implementation are described in chapters 4 and 5.

All the tests were performed with two Android phones, the specifications for each device are in table 3.1, the two devices have different characteristics in order to test the protocol on different Android versions and hardware, since we designed the testbed and the protocol with standardized characteristics, this is possible.

Sony Xperia LT30p	Android 4.3 Jelly Bean Dual core Qualcomm Krait MSM8260-A de 1.5 GHz processor 1GB of RAM
Sony Xperia M4Aqua	Android 5.0 Lollipop Octa-core Qualcomm MSM8939 a 1.5 GHz processor 2GB of RAM

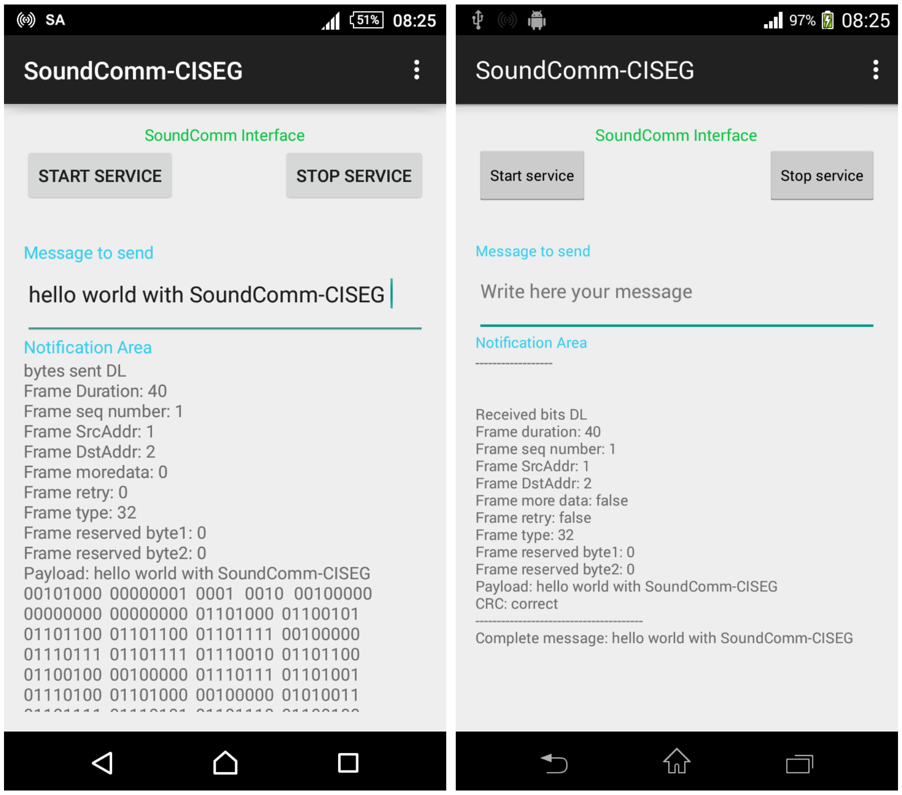
Table 3.1: Hardware used in the tests.

The testbed was installed in both phones, the testbed can act as a transmitter or receiver, depending on the user input, if there is no message on the text box when we start the service, the testbed automatically acts as a receiver, if not, it acts as a transmitter. An example of the operation of sending a message is shown in figure 3.2, in all the experiments the transmitter speaker is aligned with the receiver microphone. The tests presented in this chapter aims to validate the protocol parameters, on all the experiments the distance of transmission is 5 cm, and the volume used is set to the half of the volume capacity being 8 from the 15 standardized possible values of volume on Android, an example of the device positioning for the tests is shown in figure 3.3.

3.5 SoundComm-CISEG Architecture

As we mentioned in previous sections, the protocol is based on the IEEE 802 reference model, but specifically it uses the structure of the IEEE 802.11 [7]. The proposed protocols were thought for simplex and point to point communication, having this considerations in mind, it is possible to propose similar layers and sublayers as in the IEEE 802.11 standard with one difference, the data-link layer operates just as unacknowledged connectionless-mode service. SoundComm-CISEG is comprised by two protocols due their two layers, physical and data-link, and these layers are also divided on sublayers that perform different tasks in order to make easier the process of abstraction and improve the modularity.

The main structure of SoundComm-CISEG is shown in figure 3.4. As



hello world with SoundComm-CISEG

```
bytes sent DL
Frame Duration: 40
Frame seq number: 1
Frame SrcAddr: 1
Frame DstAddr: 2
Frame moredata: 0
Frame retry: 0
Frame type: 32
Frame reserved byte1: 0
Frame reserved byte2: 0
Payload: hello world with SoundComm-CISEG
00101000 00000001 0001  0010  00100000
00000000 00000000 01101000 01100101
01101100 01101100 01101111 00100000
01110111 01101111 01110010 01101100
01100100 00100000 01110111 01101001
01110100 01101000 00100000 01010011
```

```
------------------

Received bits DL
Frame duration: 40
Frame seq number: 1
Frame SrcAddr: 1
Frame DstAddr: 2
Frame more data: false
Frame retry: false
Frame type: 32
Frame reserved byte1: 0
Frame reserved byte2: 0
Payload: hello world with SoundComm-CISEG
CRC: correct
----------------------------------------
Complete message: hello world with SoundComm-CISEG
```

Figure 3.2: SoundComm-CISEG testbed as a transmitter (left) and as a receiver (right).

can be seen in the figure 3.4, all the sublayers have their own SAPs to exchange data between them. There are two SAPs on each sublayer using one for transmission and other for reception, the data exchanged via the SAPs has the structure of PDUs which were thought as an implementation independent data exchange format for standardization purposes, the data flows are slightly different in the case of transmission and reception, these flows are illustrated in the figures 3.5 and 3.6. The details of every sublayer are described in sections 3.6 and 3.7.

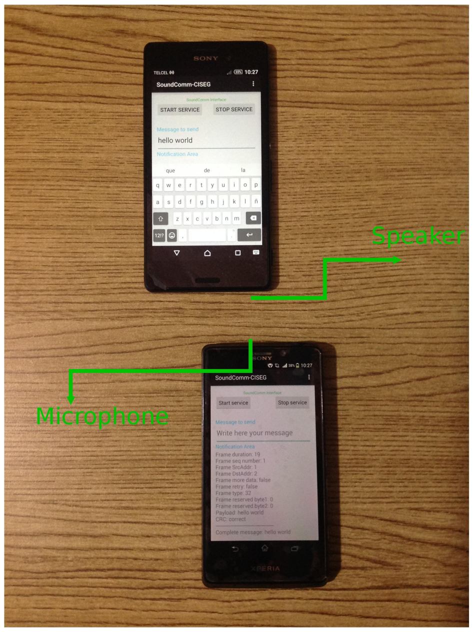


Figure 3.3: Positioning of the devices for the design tests.

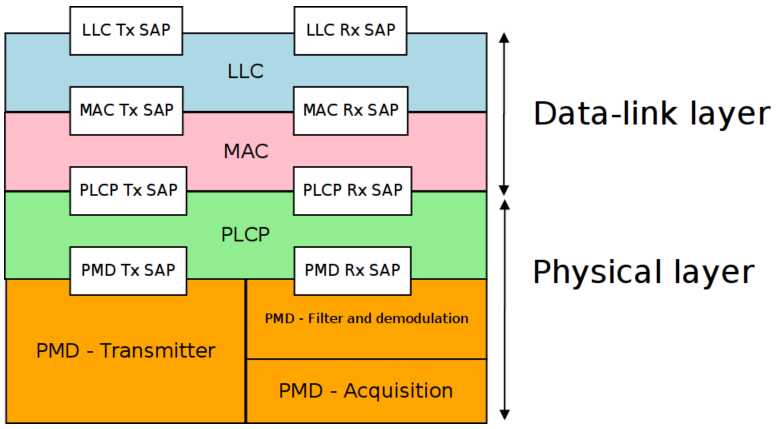


Figure 3.4: SoundComm-CISEG architecture.

3.6 Physical layer

3.6.1 Physical Medium Dependent sublayer

The physical medium dependent sublayer (PMD) is a sublayer defined for the interaction with the transmission medium, the acoustic signals are pro-

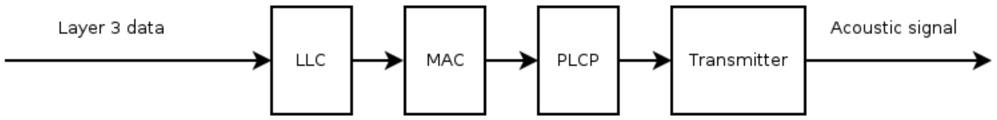


Figure 3.5: Data flow in the case of transmission.

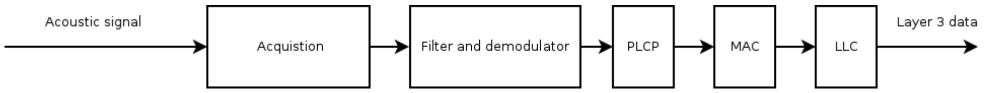


Figure 3.6: Data flow in the case of reception.

cessed in order to perform the modulation and demodulation. This sublayer has different behaviors for transmission and reception, in Rx mode it receives the sound signal from the air and delivers physical layer PDUs (PLPDUs) to the physical layer convergence procedure (PLCP), in TX mode, the PLCP delivers PLPDUs to the transmitter and it sends the PDU to the transmission medium.

3.6.1.1 Transmitter

In order to accomplish the design requirements the modulation performed on the transmitter reflects various aspects of it. In order to achieve high transmission rates with the existent hardware, the highest sample rate available on all devices was chosen for compatibility purposes [15]. The modulation chosen for the protocol was a binary FSK, other modulation techniques require more processing and modules in the receiver side, a characteristic we want to avoid due the resource constraints of the android platform. The binary FSK uses the highest frequencies on which the error rate was acceptable. Using high frequencies has the advantage of more stealthiness since higher frequencies are less perceptible to the human ear, but they tend to suffer more attenuation. The frequencies chosen where selected in function of this trade off having in mind a good transmission range. The parameters used on the transmitter are shown in table 3.2. The values of samples format and the sampling frequency are standardized values on the android platform and since the protocol is designed at API level, they cannot be modified.

The bit period was chosen experimentally and validated through the testbed. In order to define the appropriate bitrate to establish communication, several bit periods where chosen, starting from 50 samples to 150, shorter periods of the bit signal increase the bitrate and longer ones decrease the bitrate. Twenty frames where sent with different values for the bit period, the results are in figure 3.7. As expected, shorter values do not provide an accurate identification of modulation symbols, while longer ones are more effective on this aspect, starting from periods of 90 samples the demodulation is accurate, this value of 90 samples was chosen since it provides the highest bitrate available.

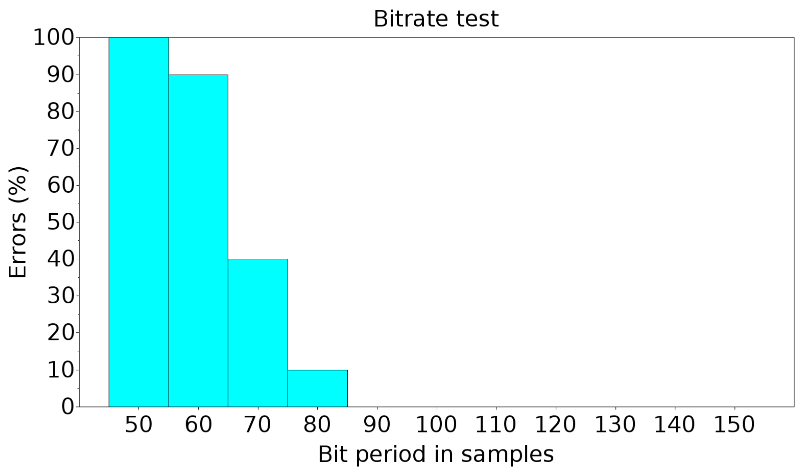


Figure 3.7: Bitrate test: Error rate depending of the size of the modulation symbol in samples.

Sound samples format	16-bit linear PCM
Sampling frequency	44100 Hz
Frequencies for FSK	18087 Hz and 19035 Hz
Bit period (T) in samples	90
Bit period (T) in seconds	0.002040816 s
Bit rate	490.0000784 bits/s

Table 3.2: Transmitter parameters

3.6.1.2 Acquisition

When a device is operating as receiver it is necessary to process the signal to get Physical layer PDUs (PLPDUs) from it, the first step in this procedure is the acquisition of acoustic data, the characteristics of acquisition of the data samples are shown in table 3.3. In a similar fashion that the transmitter these values are standardized on the android platform and cannot be modified.

Sound samples format	16-bit linear PCM
Sampling frequency	44100 Hz

Table 3.3: Acquisition parameters

Both specifications are the best available on all android devices, as stated in the Android compatibility definition document [15]. In the case of sampling frequency, higher rates are available but they are not standardized.

3.6.1.3 Filter and demodulation of the signal

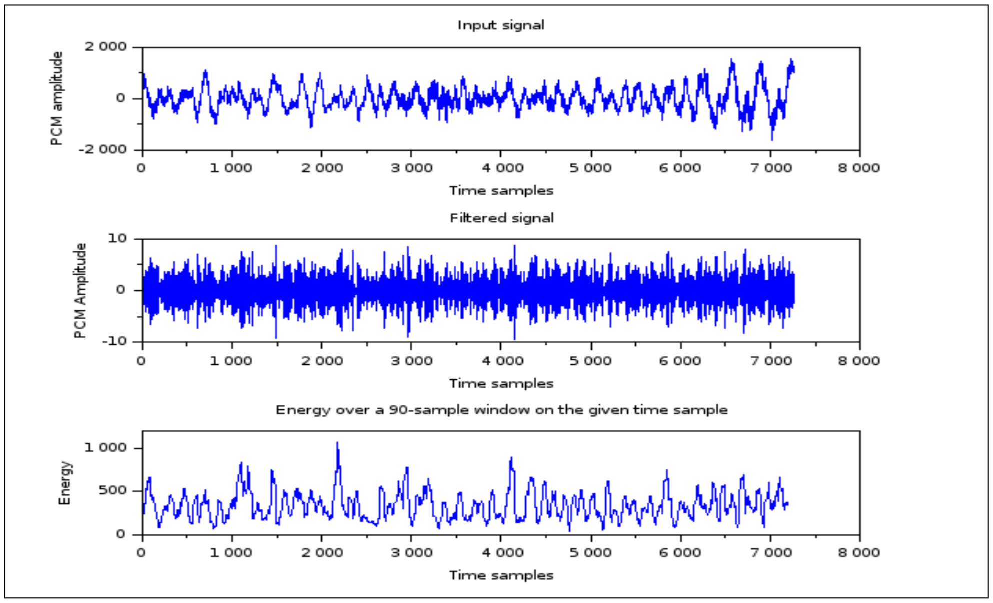
Once having the sound samples they have to be processed to make sense of them, the first step is the filter. The filter makes the protocol more resilient to ambient noise, addressing other design requirement. The characteristics for the filter are in table 3.4. These values were chosen in order to save memory, a filter with more taps is possible but it implies more memory consumption.

Type	High-pass
Number of taps	131
Cut frequency	17571 Hz

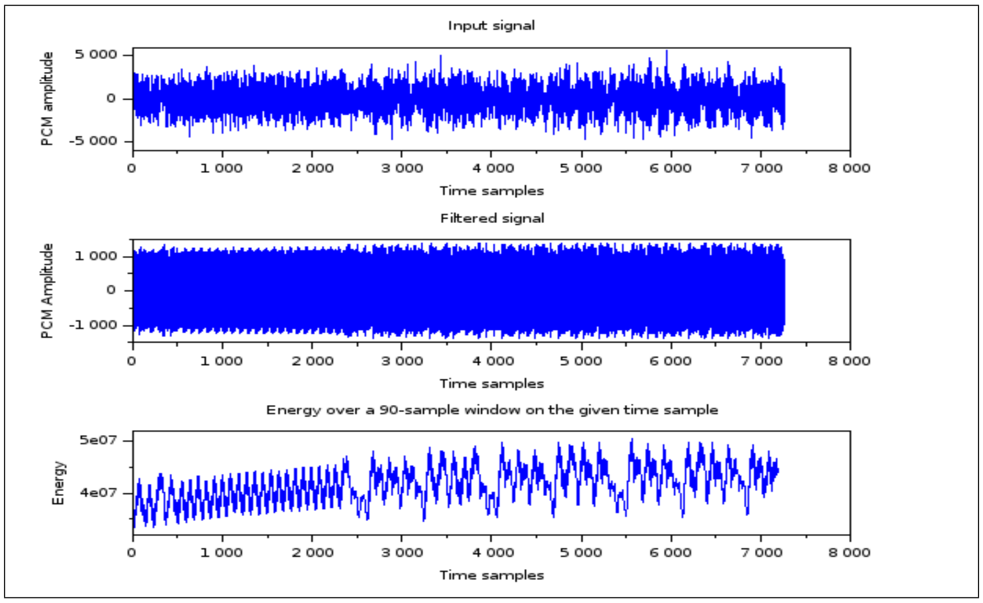
Table 3.4: Filter parameters

Due the limited resources of Android devices, the demodulation processing is performed only in the presence of carrier. Two parameter thresholds are used to detect the carrier signal, the amplitude and the energy in a 90-sample window, which is the duration of a symbol, this validation is performed for each bit demodulation. Having in consideration these two parameters is possible to detect the carrier and perform the demodulation, an example of the values taken by the energy and the amplitude is shown in figure 3.8. Experimentally the thresholds defined in table 3.5 allow the demodulation of the signal.

The demodulation method was chosen having in mind the limited memory resources in the Android devices, so a simple method for FSK demodulation is used, it is called frequency tracking. Basically, a FFT is computed



(a) Energy of the signal in the absence of carrier



(b) Energy of the signal in the presence of carrier

Figure 3.8: Energy behavior of the signal in different cases.

over the interval of a bit and the components corresponding to the modulated frequencies are compared in order to identify a 0 or a 1. The FFT size chosen is 512 samples, this value was chosen since it is the smallest necessary FFT size to demodulate the signal, it is useful for resource saving purposes, other values were tested but larger FFT sizes consume more memory and produce larger processing times. The parameters for demodulation are shown in table 3.5.

Amplitude threshold	90 in format 16-bit PCM
Energy threshold	1×10^4
FFT size for frequency tracking	512

Table 3.5: Demodulation parameters

3.6.2 Physical Layer Convergence Procedure sublayer

The physical layer convergence procedure sublayer (PLCP) acts as an interface to the data-link layer. The primitives issued by the MAC sublayer are executed at the PLCP, it generates and processes the PLPDUs and also coordinates the actions of the PMD modules. These functionalities are explained in the next sections.

3.6.2.1 PMD coordination

In the Android platform several functionalities are only accessible via asynchronous APIs, so at user level the initialization and task times are not directly controllable, a method to address this aspect is the coordination at PLCP. The PLCP is responsible for coordinating the sequence of activation, deactivation and the cooperation of the PMD modules (i.e. acquisition, demodulator and transmitter). The PLCP offers two modes of operation, Tx mode for transmission and Rx mode for reception, these modes are implicitly initiated with the primitives issued at the data-link layer.

3.6.2.2 PLPDU generation and processing

When the MAC PDUs are received by the PLCP, they are processed in order to create a PLPDU, on the receptor the PLPDUs are received and the MAC content is delivered to the correspondent sublayer. A value important to know is the best size of the physical layer MTU where synchronization at physical layer is achieved, we tested different values from 50 samples to

150, the error rate is reported in figure 3.9, starting from sizes of 120 bytes the synchronization is lost and the errors occur, shorter sizes are viable for reliable communication. We have chosen an MTU of 110 bytes which is the maximum size where synchronization is achieved in order to maximize the size of the data in every PLPDU.

The structure of the PLPDU is shown in figure 3.10. The fields are described next.

- **Preamble:** 0101010101010101 - is a sequence used for synchronization at physical layer, the pattern is similar to the used in the IEEE 802.11 standard, in this case instead of 10 octets we only use 2 due the limited bitrate of the channel.

- **Start frame delimiter (SFD):** 10111101 - is a sequence used for delimitation of the frames to be delivered at the MAC sublayer. Similar to the used in IEEE 802.11 only one octet is used due to the limited bitrate of the channel.

- **MAC PDU:** The content of the PLPDU is the frame to be delivered at the MAC sublayer. The frames are of variable size addressing other design requirement, it allows to send exactly the required bytes minimizing the transmission time, if the frames had static size, shorter messages will cause the physical layer to complete their PLPDUs with padding octets, creating longer PLPDUs to transmit which results in unnecessary long transmission times. The maximum Possible MAC size is 106 octets considering the MTU size and the control fields of the PLPDU.

3.6.3 Physical layer primitives and services specification

The primitives provide an abstraction of the interaction between layers, the physical layer primitives provide the interfaces to the data-link layer in the form of functions that abstract the details of the lower layer to upper layer. In this section we introduce the primitives used to manipulate the physical layer. The primitives at the PLCP were proposed taking as base the functionalities available via the APIs in Android since the manipulation of the hardware is done through them. The primitives and their details are shown in table 3.6.

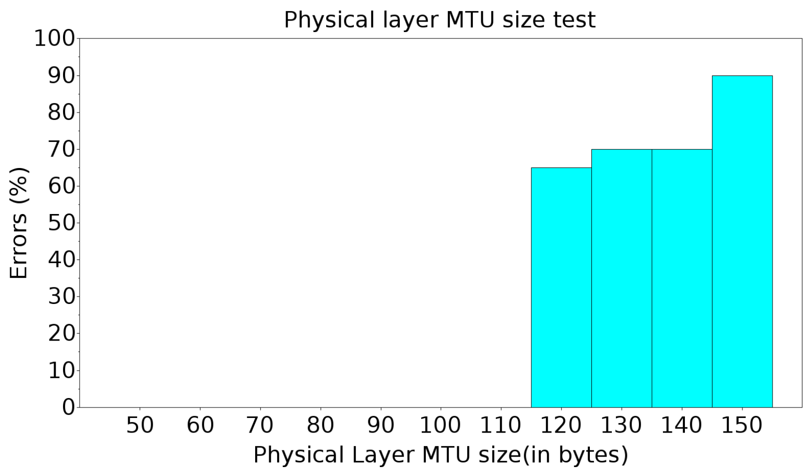


Figure 3.9: Physical layer MTU test: Error rates for every value of MTU depending of the size in bytes.

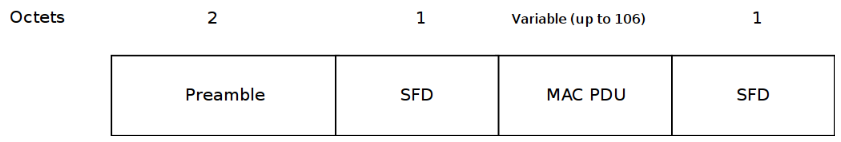


Figure 3.10: Physical layer PDU

Primitive	Details

PhyTxDataRequest	**Function:** The primitive is generated by the MAC sublayer in order to transfer data to the Tx SAP at PLCP.
	Semantics of the service primitive: The parameter data contains the MAC PDU to be transmitted.
	`PhyTxDataRequest(`
	`   Data`
	`)`

Table 3.6: PLCP primitives

Primitive	Details
	When generated: This primitive is generated when the MAC sublayer wants to allocate a Frame on the Tx PLCP SAP for transmission. **Effect of receipt:** The MAC PDU is allocated on the TX PLCP SAP.
PhyTxDataIndication	**Function:** The primitive is generated by the MAC sublayer in order to know the status of the data transferred with PhyTxDataRequest, the primitive reports whether the data was allocated for transmission or not. **Semantics of the service primitive:** The primitive has no parameters. `PhyTxDataIndication()` **When generated:** This primitive is generated when the MAC sublayer wants to know if the data transferred to PLCP is already being processed for transmission at PLCP. **Effect of receipt:** The PLCP reports the status of the data transferred with PhyTxDataRequest.

PhyTxStartRequest	**Function:** The primitive is generated by the MAC sublayer in order activate the transmission service at the PLCP.
	Semantics of the service primitive:
	The parameter of data rate is used as option to select the data rate at physical layer, at this point only one exists, but is included for extensibility purposes.
	`PhyTxStartRequest(`
	`  DataRate`
	`)`
	When generated: This primitive is generated when there is data ready to be transmitted at the Tx PLCP SAP.

Table 3.6: PLCP primitives

Primitive	**Details**
	Effect of receipt: The PLCP enters in transmission mode activating the physical layer services related with transmission and taking the data on the TX PLCP SAP for transmission.

PhyRxStartRequest	**Function:** The primitive is generated by the MAC sublayer in order to activate the physical layer services related with the reception of data.
	Semantics of the service primitive:
	The parameter of data rate is used as option to select the data rate at physical layer, at this point only one exists, but is included for extensibility purposes.
	`PhyRxStartRequest(`
	`  DataRate`
	`)`
	When generated: This primitive is generated when the MAC sublayer is prepared to receive frames. SAP.
	Effect of receipt: The PLCP activates all the modules related with reception in the physical layer.

PhyRxDataIndication	**Function:** The primitive is generated when the MAC sublayer wants to know if there is data ready to be read from the Rx PLCP SAP. **Semantics of the service primitive:** The primitive has no parameters. `PhyRxDataIndication()` **When generated:** When the MAC sublayer is ready to read data from the Rx PLCP SAP. **Effect of receipt:** The PLCP acknowledges the MAC sublayer whether there is data to be processed on the RX PLCP SAP.
PhyRxDataRequest	**Function:** The primitive is generated when the MAC sublayer wants to read a byte from the Rx PLCP SAP. **Semantics of the service primitive:** The primitive has no parameters.

Table 3.6: PLCP primitives

Primitive	Details
	`PhyRxDataRequest ()` **When generated:** When the MAC sublayer wants to read data from the Rx PLCP SAP. **Effect of receipt:** The PLCP allows read data from its buffer.
PhyRxEnd	**Function:** The primitive is generated when the MAC sublayer wants to stop the reception of data from the physical layer **Semantics of the service primitive:** The primitive has no parameters. `PhyRxEnd ()` **When generated:** When the MAC sublayer has already received all the frames, or the timeout is reached. **Effect of receipt:** The PLCP stops the modules related with reception.

Table 3.6: PLCP primitives

3.7 Data-link layer

The data-link layer comprises two sublayers the MAC sublayer and the LLC sublayer, both parts are considered but due the reach of our protocol (the condition of being simplex and point to point) several functionalities on these sublayers are trivial, despite this fact, all sublayers are designed to be extensible allowing the implementation of more complex behaviors. In this section an explanation of the data-link layer is given, in a similar fashion as given in the physical layer section.

Bits	8	8	4	4	1	1	6	8	8	Variable (up to 792)	8
	Duration	SeqNumber	SrcAddr	DstAddr	MoreData	Retry	Type	ReservedByte1	ReservedByte2	Frame Body	FCS

Figure 3.11: MAC Frame

3.7.1 MAC sublayer

As stated in chapter 1, the MAC sublayer provides frame-based and connectionless-mode data transfer, plus mechanisms for medium access, this last task is very simple in the scenario since is considered only point to point communication in simplex mode. The MAC sublayer also issues the primitives to the physical layer in order to achieve transmission and reception, this communication is frame based, so this layer is responsible to create the frames for transmission and to parse them for reception, also it is capable of segmentation if the message is larger than the maximum frame size. The frame format is shown in figure 3.11. All specified parameters were considered having in mind the limited bitrate of the physical layer specifications, but also the frame contains all the basic data to establish communication and other parameters that could be used to enable the protocol to perform more complex operations, for example half-duplex communication.

- **Duration:** Indicates the longitude of the frame in octets, having a frame of variable length, the transmission only implies to send exactly the required octets, addressing the requirement of minimizing transmission times.

- **SeqNumber:** Sequence number, indicates the number of frame, the sequence control uses this parameter to reassemble the entire message.

- **SrcAddr:** Source Address, on a point to point protocol does not make sense to use addresses since just two entities are involved, but the source address is specified due extensibility purposes.

- **DstAddr:** Destination Address. Similar to source address.

- **MoreData:** If is set to 1, indicates that there are more frames to send after this one.

- **Retry:** If set to 1, indicates that the current frame is a retransmission.

- **Type:** On this version of the protocol only one type exists, the data type, but for extensibility purposes, the type field could be used to enable the protocol to have more complex behaviors, as in IEEE 802.11 [**?**], this field has 6 bits.

- **ReservedByte1 and ReservedByte2:** Reserved bytes for extensibility purposes, considering that this protocol supports short distances, the bytes reserved are only two in order to maintain the control data small compared with the frame body size. These octets could be used by characteristics like quality of service or flow control, etc.

- **Frame body:** The effective payload of the frame, the maximum size is 99 octet and the minimum is 1 octet, the maximum transmission time with a frame of full length is $1.763265306s$.

- **FCS:** Frame check sequence, a sequence to detect errors on the frame, for this protocol is CRC-8-CCITT

All mentioned parameters try to minimize the control information in order to transmit more effective data considering the limited capacity of the channel. For example, the addresses just use 4 bits for each one, considering that the protocol is meant to be used in short distances so, in case of further multi-point communication there will be no more than 16 entities performing communication at the same time.

3.7.2 LLC sublayer

The LLC sublayer is responsible for the interaction of the data-link layer with the higher layer in a similar fashion that the PLCP, this sublayer implements the primitives to manipulate the data-link layer, also controls the logic of the communication, in this case is fairly obvious considering a simplex and point to point protocol. Other function of the LLC is to multiplex network protocols, in our case it is considered that only one protocol exists at network layer so the multiplexation is non existent and just one pair of SAPs exist between data-link and network layer. In resume the LLC is operating as a unacknowledged connectionless-mode service.

The LLC sublayer controls the logic of communication, in the scenario of a simplex and point to point protocol, we have just two peer entities interacting in a simplex communication, the flow of frames only comprises three cases:

- The frame arrives in time without errors

- The frame arrives after the timeout or never arrives

- The frame arrives with errors

These flows and their respective times are illustrated in figures 3.13a, 3.13b and 3.13c. The first flow example in figure 3.13a, shows the correct communication of two frames, the sequence numbers are designated incrementally starting with one with each new transmission flow. Since we have a simplex transmission, we are no capable of receiving ACKs in order to continue sending frames, so the time to wait $Twait$ between transmissions is calculated by the following expression:

$$Twait = Tprop + Tproc \tag{3.1}$$

Where $Tprop = distance/(340.29m/s)$, taking the speed of sound as propagation speed, to calculate the time $Tprop$ we consider different transmission distances for the protocol in table 3.7.

Distance	**Tprop**
2.5 cm	0.000073467 s
5 cm	0.000146933 s
10 cm	0.000293867 s
15 cm	0.0004408 s

Table 3.7: Propagation times depending on the distance

$Tproc$ is the processing time at the receiver, this time is implementation dependent, in this case $Twait$ should be calculated experimentally for the testbed in specific. In order to test this characteristic, a message long enough to generate 10 frames was sent, the variable $Twait$ is varied in the range of 3 to 8 seconds to measure if the link is maintained during all the transmission, for each value of $Twait$ the process was repeated 10 times, the average of the maximum frame where the link remained established is shown in figure 3.12, any $Twait$ value above 5 seconds can be used for transmission. Depending on the application, it could be useful to transmit in longer periods or shorter periods but not less than periods of one frame every 5 seconds. In this protocol we use $Twait$ as 5 seconds in order to transmit as fast as possible. $Twait$ could be reduced using devices with more powerful capabilities but the protocol aims to be usable for the majority of the Android mobile devices. With these considerations, our protocol is capable of deliver in a reliable way 99 bytes of effective payload in $Ttrans+Twait = 6.763706106s$

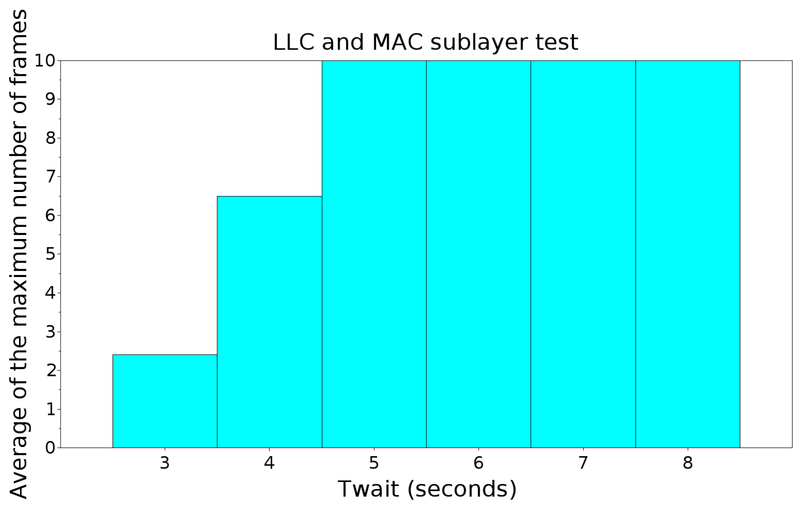


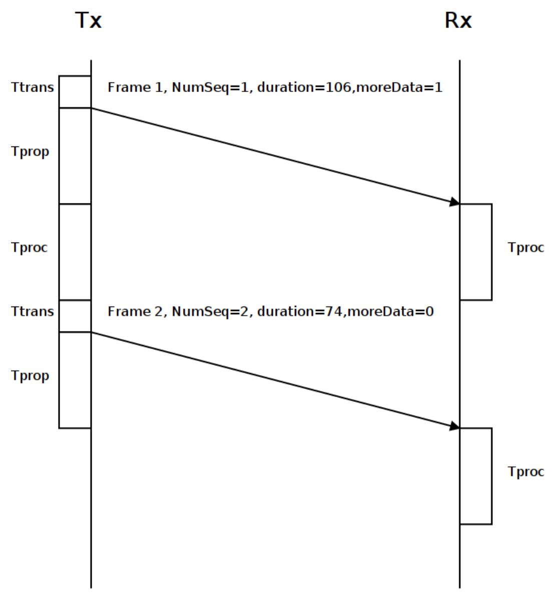
Figure 3.12: MAC frames and logical link control test: Average of the maximum number of frames on which the link is maintained.

seconds. It is important to emphasize that $Twait$ remains static due the fact that this time is only used when a client message is segmented on two or more frames, implying the transmission of frames with the maximum length, and the last frame does not require $Twait$.

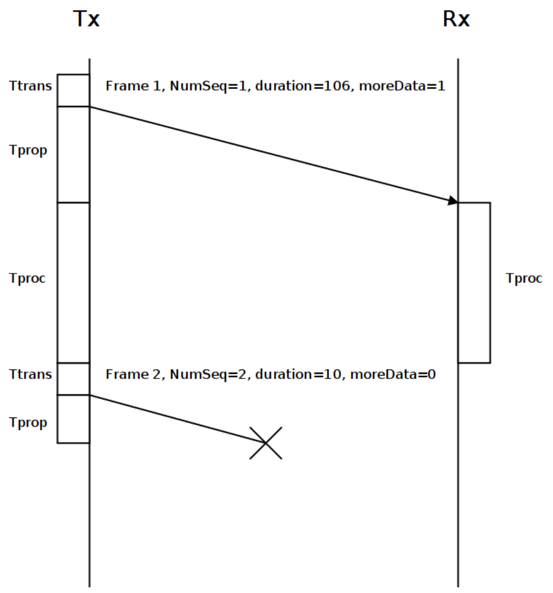
In figure 3.13b, one fault condition is stated, this condition is easily acknowledged by the receiver using the sequence numbers searching for one missing or reaching a defined timeout for reception, this timeout of 15 seconds was defined in order to take into account the miss of a frame due to excessive $Tproc$ at the receiver, in this case, the timeout allows to receive the next frames if they exist considering that the transmissions are done each 5 seconds which correspond to $Twait$. If the receiver does not receive information on this time lapse, the condition is reported to the upper layer via the primitive *LLCRxDataStatusIndication*.

The third fault condition is represented in figure 3.13c, the figure shows a delivered frame with errors, it is easily addressable due the frame check sequence, the CRC, which allow the receiver to know if the frame has no errors, in the contrary case, the errors are notified to the upper layer via the primitive *LLCRxDataStatusIndication*.

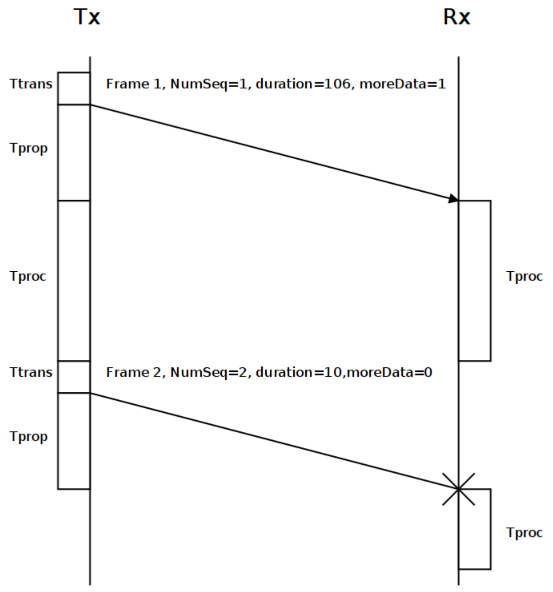
All these flow exemplify the operation of the protocol and the logic behind each transmission, as a manner of resume, all LLC parameters are


Tx
Rx
Ttrans
Frame 1, NumSeq=1, duration=106,moreData=1
Tprop
Tproc
Tproc
Ttrans
Frame 2, NumSeq=2, duration=74,moreData=0
Tprop
Tproc

(a) Transmission of two frames without errors


Tx
Rx
Ttrans
Frame 1, NumSeq=1, duration=106, moreData=1
Tprop
Tproc
Tproc
Ttrans
Frame 2, NumSeq=2, duration=10, moreData=0
Tprop

(b) Transmission of one frame with a missing frame


Tx
Rx
Ttrans
Frame 1, NumSeq=1, duration=106, moreData=1
Tprop
Tproc
Tproc
Ttrans
Frame 2, NumSeq=2, duration=10,moreData=0
Tprop
Tproc

(c) Transmission of two frames with an error in the second frame

Figure 3.13: Different flow control conditions

Timeout	15 s
Time wait between transmission of frames (*$Twait$*)	5 s
Effective bitrate	130.106185309 bits/s

Table 3.8: LLC sublayer parameters

shown in table 3.8.

3.7.3 Data-link layer primitives and services specification

In a similar fashion that the physical layer, the data-link layer also implements primitives to be manipulated, in this case by a potential network layer or an application using the data-link protocol SoundComm-CISEG, the data-link primitives are presented in this section.

Primitive	Details
LLCTxDatarequest	**Function:** The primitive is generated by the network layer in order to transfer data to the Tx SAP at LLC. **Semantics of the service primitive:** The parameter data contains the network layer PDU (NLPDU) to be transmitted, SrcAddr contains the MAC source address and DstAddr contains the MAC destination Address. `LLCTxDatarequest( msgFromNL, SrcAddr, DstAddr )` **When generated:** This primitive is generated when the Network layer wants to allocate a NLPDU on the Tx LLC SAP for transmission. **Effect of receipt:** The PDU is allocated on the TX LLC SAP.

Table 3.9: LLC primitives

Primitive	**Details**
LLCTxDataIndication	**Function:** The primitive is generated by the network layer in order to know the status of the data transferred with LLCTxDatarequest, the primitive reports whether the data was allocated for transmission or not. **Semantics of the service primitive:** The primitive has no parameters. `LLCTxDataIndication()` **When generated:** This primitive is generated when the network layer wants to know if the data transferred to LLC is already being processed for transmission at LLC. **Effect of receipt:** The LLC reports the status of the data transferred with LLCTxDatarequest.
LLCRxDataIndication	**Function:** The primitive is generated when the network layer wants to know if there is data ready to be read from the Rx LLC SAP. **Semantics of the service primitive:** The primitive has no parameters. `LLCRxDataIndication()` **When generated:** When the network layer is ready to read data from the Rx LLC SAP. **Effect of receipt:** The LLC acknowledges the network layer whether there is data to be processed on the RX LLC SAP.
LLCRxDatarequest	**Function:** The primitive is generated when the network layer wants to read a NLPDU from the Rx LLC SAP. **Semantics of the service primitive:** The primitive has no parameters.

Table 3.9: LLC primitives

Primitive	**Details**
	`LLCRxDatarequest()` **When generated:** When the network layer wants to read data from the Rx LLC SAP. **Effect of receipt:** The LLC allows read data from its buffer.
LLCRxDataStatusIndication	**Function:** The primitive is generated when the network layer wants to know whether there are errors on the received NLPDUs. **Semantics of the service primitive:** The primitive has no parameters. `LLCRxDataStatusIndication()` **When generated:** When the network layer wants to know whether there are errors on the received NLPDU. **Effect of receipt:** The LLC reports to the network layer the status of the received NLPDUs.

Table 3.9: LLC primitives

With these primitives the data-link layer is complete and fully operational, as we mentioned during this chapter, the specifications are provided in order to achieve all design requirements with the existing hardware in Android devices, making this protocol as standardized as possible and capable of running in the majority of devices. All these values were validated in an experimental way, in the next chapters we present the design of our testbed who validated our protocol specifications.

Chapter 4

Testbed: Physical layer

In order to validate the specifications of the SoundComm-CISEG protocols, a testbed was implemented, the presentation of the testbed is divided in two layers, the physical and the data-link; first, is presented how the physical layer performs its functionalities. As stated in chapter 1, the work takes as reference the IEEE 802 model. The implementation on Android mobile devices is largely dependent of how the APIs and hardware functionalities are provided, this is particularly important since the physical layer controls and coordinates the interactions with the microphone and the speaker, which at the user level, can be utilized only via APIs provided by the platform itself.

Several considerations have been taken in terms of a realistic implementation, Android devices are usually resource constrained so, complex modulations and demodulations procedures were avoided, in this case a binary FSK was chosen for modulation and the method for demodulation is frequency tracking. Other aspect is the memory management, all variables were preallocated and configured as *static* i.e. the use of memory remains constant and it does not grow, additionally all computations were performed on arrays of 512 double values in size, other values were tested but larger arrays occupied all the memory and showed worst performance; in contrast smaller arrays provided not enough resolution on the FFT computation (which is used in frequency tracking) in order to achieve a correct demodulation.

The testbed implements the protocol structure, so the architectural elements are the same as in the figures 3.4, 3.5 and 3.6, each architectural element is implemented in the form of an asynchronous thread at user level programed in java, every thread except for the acquisition is provided by one or more SAPs in the form of arrays or lists, working at the API level has certain constraints addressed by this testbed, for example, at the user level all applications are isolated inside a container which has limited memory (approximately 10 Mb in the tested phones), additionally the access to the hardware is restricted and is only accessible via APIs. All these facts impacts on the protocol performance since it could be faster to operate the hardware at lower level without programming abstractions, as matter of resume the presented testbed works with the capabilities of every Android application at the user level without using any special privilege but the permissions for access to the speaker and the microphone.

4.1 PMD sublayer

The PMD is a sublayer which is concerned with how the signals are processed in order to deliver a service, on this testbed, three threads were used to implement the PMD, the implementation of these three procedures is explained in next subsections.

4.1.1 Acquisition

The module of acquisition has the task of getting the audio samples from the microphone and put them on a buffer that can be read by the demodulator block. The acquisition in Android is performed via an API called AudioRecord. This API allows to retrieve the sound samples directly instead of recording the data to a file, like in the case of other API called MediaRecorder, the object used to record data was instantiated as shown in listing 4.1

```
Acqminbuffer_size=AudioRecord.getMinBufferSize(
  44100,
  AudioFormat.CHANNEL_IN_MONO,
  AudioFormat.ENCODING_PCM_16BIT
  );

AudioRecord obj=new AudioRecord(
  MediaRecorder.AudioSource.MIC,
  44100,
  AudioFormat.CHANNEL_IN_MONO,
  AudioFormat.ENCODING_PCM_16BIT,
  Acqminbuffer_size
  );
```

Listing 4.1: AudioRecord object instantiation

The first function is used to determine the size of the recording buffer, this size cannot be fixed since its a different value for each device, and should be determined via this function, additional parameters are the sampling frequency, in this case $44100Hz$, several sampling frequencies are supported by android, but $44100Hz$ is the higher guaranteed to work on every device according to the Android standards [15]. It is important to choose a high sampling frequency because it allows higher transmission rates.

The next parameter is the configuration of channels, for the detection and processing only one channel is required. The enconding defines the resolution of the samples retrieved and how the levels of quantification are distributed on the analog to digital converters, in the case of 16 bit PCM, the quantization levels are distributed uniformly and the values correspond to the range of a type *short* variable in java, which is the data type used in the code.

Once initialized the object it starts to retrieve samples on demand in an array of size $Acqminbuffer\_size/2$. Since the buffer size is different from device to device, one way to manipulate this data in a device-independent way is to allocate the samples in a different buffer of a fixed size, and then the next block, the filter, reads the data from this buffer of fixed size [16].

4.1.2 Filter and demodulation

4.1.2.1 Filter

Having allocated the received sound samples in the buffer of fixed length, the second block starts to process the data in a window of 382 samples at a time. The process of filtering involves the process of convolution between the input signal and the filter coefficients. The filter used is a Finite impulse response (FIR filter) with 131 taps, this filter is a high-pass with a cut frequency of $17518Hz$. The convolution is defined for a Q-tap FIR filter by the following equation:

$$y(n) = \sum_{k=0}^{Q-1} h(k)x(n-k) = h(k) * x(n) \tag{4.1}$$

Where $h(k)$ are the filter samples, and $x(n)$ the input data, an optimization can be done using the following property of convolution, called convolution theorem:

$$h(k) * x(k) = IDFT(H(m) \cdot X(m)) \tag{4.2}$$

similarly

$$DFT(h(k) * x(k)) = H(m) \cdot X(m) \tag{4.3}$$

Where IDFT denotes the inverse discrete Fourier transform, and DFT the discrete Fourier transform, being H(m) the DFT of h(k) and X(m) the DFT of x(k) respectively. Having this property in consideration, we could compute the convolution using the fast Fourier transform (FFT) of both input sequences, multiplying between them, and compute the inverse FFT, this method is used on the testbed.

As we mentioned at the beginning of this section, the filtering is done over a window of 382 samples in real time, to accomplish this, the filtering is performed by parts in the form of small arrays which are manipulated with the method of *overlap and add*. Computing the filter in real time has advantages compared with a one-time filtering. First, the input signal is filtered as soon as it arrives and only the part that is needed is processed, otherwise it would be necessary to wait for the entire signal to process it, even if the usable data is contained in the first 10% of the recording, causing a considerable delay by computing the following 90% of irrelevant data. Second, the one-time filtering needs bigger arrays to be computed, which implies more memory consumption, a non-desirable characteristic on our scenario.

The overlap and add method, as showed in figure 4.1, is designed to provide a way for filtering in smaller arrays, decomposing a large sequence into parts which are filtered one by one, and the result of each part is added to the last result, at the end, the complete filtered signal is obtained. Implementing this method requires to fulfill the following property, if $h(k)$ is of length Q and $x(n)$ is of length P, the length L of the final sequence $y(n)$ will be:

$$L = Q + P - 1 \tag{4.4}$$

In this work Q and P were selected having as reference the performance with values which make L power of two in order to use the FFT algorithm. As a consequence it is possible to implement a fast convolution using FFTs. In this implementation the following values were selected: $Q = 131$ and $P = 382$, then the integer M is calculated, $M = L - (Q - 1)$, this value of

M indicates how many samples of the array will be concatenated in order to get the entire filtered signal.

Having selected the values for Q and P, in order to start the processing of the first part of input data, both sequences, filter samples and the input data are completed with zeros until both have L samples, the filtering procedure which makes use of FFTs and IFFT is applied, the first M samples of the result are taken as valid, the remaining samples are added to the first input samples of the next part of the input data to be computed. At the end, the result is the concatenation of the first M samples in each iteration, this process is shown in figure 4.1.

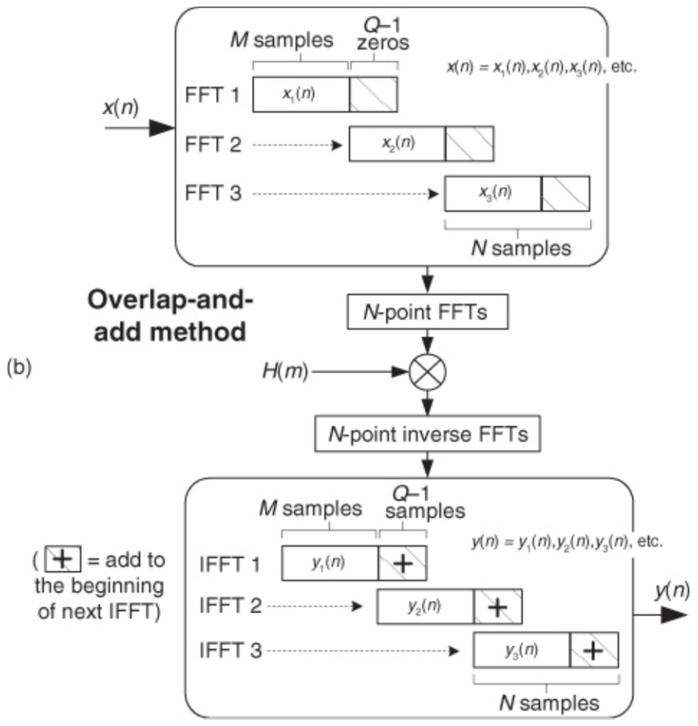


Figure 4.1: Overlap and Add method for filtering [3].

4.1.2.2 Demodulation

The process of filtering and demodulation are done in the same block, and in terms of implementation, the same thread, as soon as the samples are filtered, they proceed to be demodulated. After getting every chunk of the filtered signal, the process of carrier sensing is applied, the process takes in consideration two parameters, the amplitude and the energy of the signal in a window of $T = 90$ samples, which is the period of a symbol.

The mentioned parameters are used to compute the process of demodulation selectively, having a resource constrained platform, it is important for the performance. Both parameters worked in function of a threshold defined experimentally. First, in the case of amplitude, a value of amplitude $A = 90$, corresponding to the 16 bit PCM sample is detected in the filtered signal, then a validation through the energy of the signal on the given window proceeds, the energy is given by the next expression:

$$E_s = \sum_{i=1}^{T} |S[n]|^2 \tag{4.5}$$

The requisite of energy of a given chunk of size T to be processed is $E_s > 1 \times 10^4$. If the conditions of amplitude and energy are fulfilled, the data is demodulated using frequency tracking, basically a FFT over the data is computed, and a check for the maximum component on the magnitude spectrum is performed, with this procedure, the signal can be transformed to a bit array which is processed in the PLCP for further delivery to data-link layer via the PMD Rx SAP.

4.1.3 Transmitter

Mobile devices usually are resource limited in terms of memory and processing, due to these limitations the communication parameters were selected, in this case, a simple modulation scheme was chosen, a binary frequency modulation. The modulation is performed sending two tones to the speaker, depending if we want to send a 1 or a 0.

Android provides an API capable of sending raw 16 bit PCM data to the speaker in order to be played, it is called AudioTrack, the instantiation of the object is presented in the listing 4.2.

```
buffer_size=AudioTrack.getMinBufferSize(
  44100,
  AudioFormat.CHANNEL_OUT_MONO,
  AudioFormat.ENCODING_PCM_16BIT
  );

AudioTrack obj=new AudioTrack(
  AudioManager.STREAM_MUSIC,
44100,
AudioFormat.CHANNEL_OUT_MONO,
AudioFormat.ENCODING_PCM_16BIT,
buffer_size ,AudioTrack.MODE_STREAM
);
```

Listing 4.2: AudioTrack object instantiation

The parameter for transmission are analogous to acquisition parameters, in a similar fashion, they define how the object will operate, but, for transmission instead of acquisition. The only different parameter is the operation mode, static or stream, on static mode the sound samples are allocated in memory and then played, on stream mode, the samples are directly written to the audio buffer saving memory in the process, being the last option better in this case [17].

The modulation is generated via two arrays containing cosine data for both frequencies, these arrays are later provided to the AudioTrack object and it plays the modulated signal in the form of audio. In order to known which cosine signal send to the air, the physical layer PDU generated by the PLCP (which is allocated on the PMD Tx SAP) is parsed and the AudioTrack object acts in response.

4.2 PLCP sublayer

This sublayer is inspired in the IEEE 802.11 specification [**?**]. The main purpose of this layer is to generate Physical layer PDUs from the MAC layer PDUs in the next higher layer, in this protocol, this layer also performs the work of coordination for the physical layer procedures.

The PLCP is a sublayer of interaction between low level physical layer functionalities and the data-link layer, the PLCP is responsible for the coordination between them and it performs the invocation of primitives. First we have to consider that at user level many tasks on physical layer are performed via APIs (i.e. Acquisition of data from the environment and transmission of data to the speakers), these APIs are asynchronous procedures not directly controllable, so the coordination of activation, deactivation, start and stop of procedures invoked from data-link layer must be regulated, the PLCP implements these mechanisms with the existent tools at user level (flags and functions to monitor the tasks on APIs, etc.).

The other purpose is to encapsulate the MACPDU to a PLPDU, the format of the PDUs transmitted between layers in terms of implementation is a java type char[].

	Input(data type)	**Output**(data type)
PLCP	MAC PDUs (binary String)	Physical layer PDUs (char array)
Transmitter	Physical layer PDUs (char array)	16 bit PCM samples (short array)

Table 4.1: Physical layer inputs and outputs on Tx mode

	Input(data type)	**Output**(data type)
Acquisition	16 bit PCM samples (short array of a device dependent length)	16 bit PCM samples (short array of a fixed length)
Filter and demodulation	16 bit PCM samples (short array of a fixed length)	Physical layer PDUs (char array)
PLCP	Physical layer PDUs (char array)	MAC PDUs (binary string)

Table 4.2: Physical layer inputs and outputs on Rx mode

A resume of the input and output data formats in each block on physical layer on both modes, transmission and reception, can be seen in the tables 4.1 and 4.2.

In this chapter the explanation of how the physical layer operates in the testbed is given, the implementation of the testbed aims to be resource saving in order to be executed in the majority of android devices. In the next chapter an explanation of the data-link testbed is given to understand the complete functionality of the testbed.

Chapter 5

Testbed: Data-link layer

The data-link layer is composed by two sublayers, the MAC sublayer and the LLC sublayer. The MAC sublayer receives and sends MAC PDUs from and to the physical layer, and the LLC primarily provides the multiplexation of protocols at the data-link layer and the logic of the communication with peer entities, in this testbed, the MAC sublayer is fully implemented for the case of a simplex and point to point protocol while the LLC has certain differences, as we mentioned in chapter 3, working on a mobile device, and specifically with hardware not designed for communication establishment, has certain limitations, due to this reason some security functionalities of the LLC sublayer are not implemented on the testbed, the multiplexing at LLC is trivial due the characteristics of our protocol, and the link control is delimited by the same characteristics of the protocol proposed, it is simplex and point to point.

5.1 Input data manipulation

All the testbed works with binary data internally, but for demonstration purposes the data to be sent is introduced via a graphical interface with a textbox, this data is transformed into binary strings to be threated as bits for the functions at data-link layer, this transformation is done with the instruction in the listing 5.1.

```
byteToSend=Integer.toBinaryString(data.charAt(i));
```

Listing 5.1: Text to binary string conversion.

The given instruction can transform every char at the input string, to a string composed by ones and zeros which are the binary representation of every character. Similar strings are used for data transfer at SAPs. On reception the inverse transformation is performed in order to make sense of the parameters parsed as shown in the listing 5.2.

```
rxFrameSeqNumber = Integer.parseInt(byte2parse, 2);
```

Listing 5.2: Binary string to text conversion.

5.2 MAC sublayer

The MAC sublayer performs the interaction with the physical layer, on the testbed, the data-link thread issues the primitives to control the PLCP thread. On Tx mode, this sublayer fills the PLCP Tx SAP and then it uses the primitives to start transmission, in a similar fashion on Rx mode, the MAC sublayer waits for the notification of data to read at the PLCP Rx SAP, then issues the primitives to transfer data from the PLCP to the MAC sublayer.

This sublayer performs the segmentation of the LLC PDU in function of the content, the LLC sends the message in the form of a string, this string is parsed, the escape bytes are added if needed, then the MAC sublayer divides the message, creates the frames, and then they are sent to the Tx SAP at PLCP. On reception when the data from the PLCP is received, the values of the header are parsed and the incoming bytes are processed accordingly. These frames at this level are in the form of binary strings, so it makes easier to compare and convert values in order to process them, in Java, all these functions already exist.

5.2.1 Frame transmission

The MAC frames are created as binary strings, as we already have explained, this process is invoked via a function called frameConstructor as shown in the listing 5.3:

```
static void frameConstructor(
  String data,
  int seqNumber,
  int srcAddr,
  int dstAddr,
boolean moreData,
boolean retry,
int type
);
```

Listing 5.3: Java function to generate frames.

This function takes as parameters all pertinent data in their native types, it can be noticed that certain parameters are not included as parameters of the function, e.g., the duration and the CRC, these values can be inferred or computed from the other parameters so they are not included on the invocation but the processing is done in the function.

Having passed the parameters to the function, all parameters of the frame are transformed in a binary string form, and finally it can be send through the primitives shown in the listing 5.4.

```
PhyTxDataRequest(frameToSend);
PhyTxStartRequest();
```

Listing 5.4: Primitives used for transmission in the java code.

5.2.2 Frame reception

The process of reception can be seen as the inverse of transmission, the reception of data at MAC sublayer is invoked by the primitive shown in the listing 5.5.

```
PhyRxStartRequest();
```

Listing 5.5: Primitive used for reception in java code.

Once initialized the process of reception, the PLCP is responsible to notify the MAC sublayer the existence of data to be read, the MAC sublayer reads the data in octets from the PLCP Rx SAP, and performs the transformation from binary strings to integers or characters depending of the content, this process is automated within the function in the listing 5.6

```
processFrame();
```

Listing 5.6: Java function for frame processing.

5.3 LLC sublayer

The LLC sublayer is responsible for network protocol multiplexation/demultiplexation and the logic of communication, in this case for future compatibility purposes, we consider only the existence of our protocol and one network protocol, so the multiplexation is inexistent. As we mentioned in chapter 1, the modes of operation of the LLC offer different functionalities, in the testbed only the unacknowledged connectionless-mode is implemented considering the characteristics of the protocol of being simplex and point to point.

Considering the hardware limitations of the platform, the LLC and the MAC sublayers are implemented on the same thread, with a division in logic in order to maintain the coherence with our reference model. The LLC interacts directly with the next higher layer client, so it handles the primitives issued for an application or a network protocol.

In the MAC sublayer the primitives are concerned in activation and deactivation of certain services, on the LLC the process is more abstracted reducing them to primitives for data transfer, notification and status of the data received, the declaration of the primitives in code is shown in the listing 5.7

```
LLCRxDataIndication();
LLCRxDataStatusIndication();
LLCRxDatarequest();
LLCTxDatarequest();
LLCTxDataIndication();
```

Listing 5.7: Declaration of the primitives in code.

The LLC controls how and when the actions of the MAC sublayer happen, for example, the primitive *LLCRxDatarequest*, triggers the process reading the MAC Rx SAP, and controls the temporal events in order to retrieve the complete message in an automated way. A resume of the format of inputs and outputs is shown in tables 5.1 and 5.2.

In this chapter the explanation of the implementation of the data-link layer is given, all the primitives were implemented in the form of java functions. In the next chapter the tests used to acknowledge certain characteristics of operation of the soundComm-CISEG protocol are presented.

	Input(data type)	**Output**(data type)
LLC	network layer PDUs (String)	LLC PDUs (binary string)
MAC	LLC PDUs (binary string)	MAC PDUs (binary string)

Table 5.1: Data-link layer inputs and outputs on Tx mode

	Input(data type)	**Output**(data type)
MAC	MAC PDUs (binary string)	LLC PDUs (binary string)
LLC	LLC PDUs (binary string)	network layer PDUs (String)

Table 5.2: Data-link layer inputs and outputs on Rx mode

Chapter 6

Functional tests

In previous chapters the protocol specifications and the testbed were presented. The feasibility of establish communication via a data-link protocol was proved with the design tests, in this chapter, other characteristics of the protocol are tested in order to acknowledge the behavior of the protocol in different environmental conditions. This tests aim to show how the protocol parameters can be modified in order to be adapted for different usages as stated in chapter 3, in these scenarios different specifications could be possible. In resume this section shows the behavior of the protocol in different conditions in order to provide a guide of the general performance considering different values in each modifiable characteristic of the protocol.

6.1 Transmission power test

The maximum transmission distance depends on the transmission power, this parameter is controlled by the volume levels, on Android 15 volume levels are available. The 15 levels where tested and the error rates were reported, an individual frame of 110 bytes is sent, this is repeated twenty times for each sound level and for distances of 2.5 cm, 5 cm, 10 cm and 15 cm. Two variations of this experiment were done, one holding the devices on the air and the other placing the phones on a table, these two scenarios were though as possible conditions of operation. The results of these experiments are presented in figures 6.1 and 6.2 which are significantly different.

On first place, the experiments with the devices on the table can reach further distances due to the effect of the solid medium, this is particularly true on the case of 15 cm where the experiment holding the devices on the air shows that the communication is impossible, in contrast the other experiment on the table can perfectly establish communication. In shorter distances the behavior between the two scenarios is similar. These characteristics will model the transmission power on every application, different sound levels can be set depending on the intended communication distance. In further experiments we took just one distance and volume level as reference, 5 cm and volume level 8 respectively, which are the optimal parameters for a communication on the range of 5 cm on both scenarios, additionally both devices are hold in the air.

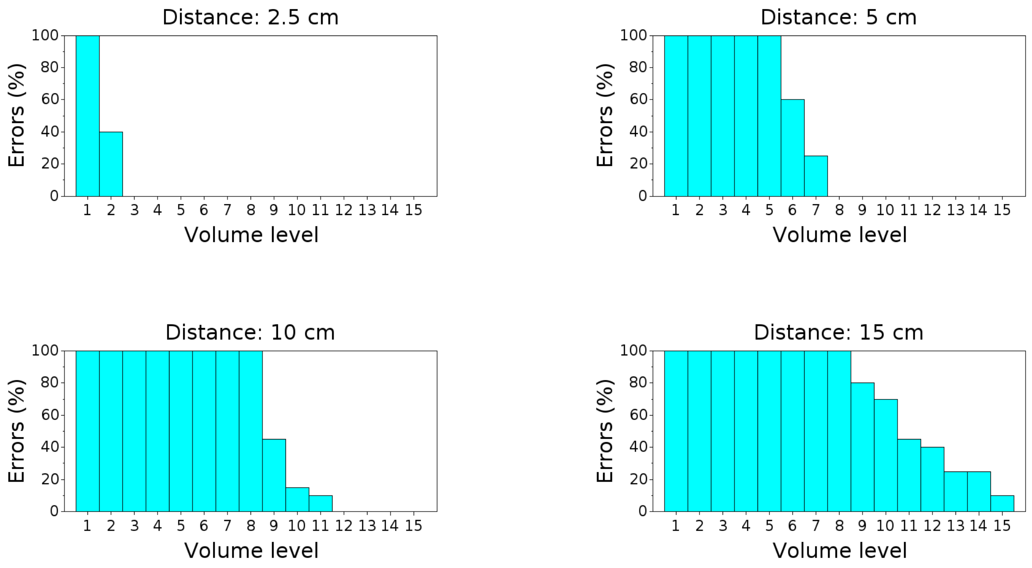


Figure 6.1: Transmission power test: Error rate for communication devices being hold in the air depending on the volume level.

6.2 Filter performance on different environments

The protocol must have certain level of resilience to ambient noise, to test that conditions twenty PLPDUs of the maximum size where sent in a room with twenty people talking and with some ambient music. The other scenario is a jammer generated with a wav file and the Scilab function *noisegen* [18] using the parameter of standard deviation as 1. This jammer was placed at 5.5 cm from each communication device, the results are interesting, the

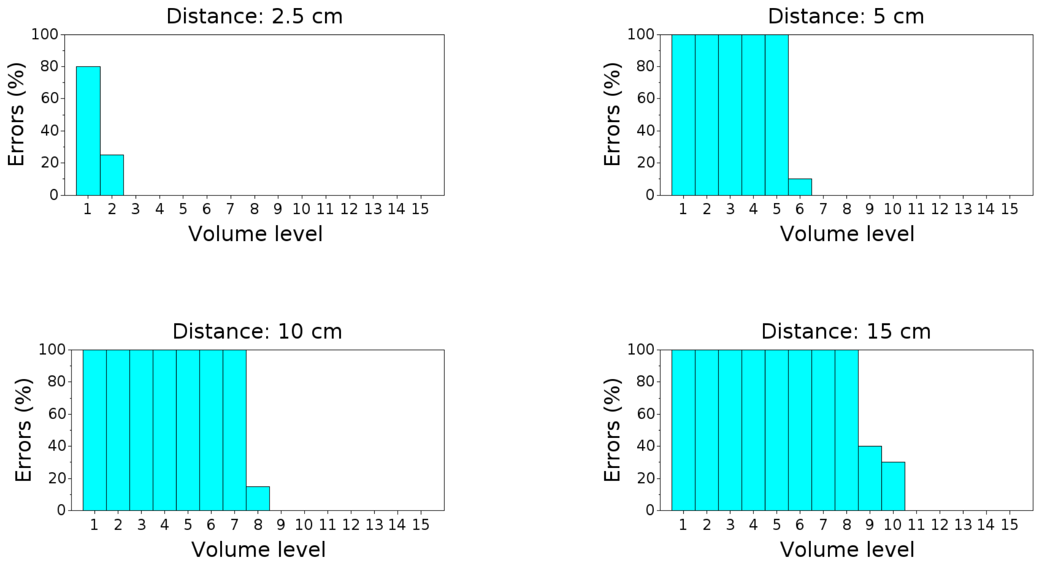


Figure 6.2: Transmission power test: Error rate for communication devices being placed on a table depending on the volume level.

Environmental conditions	**Errors (%)**
Crowded room	0
Jammer	100

Table 6.1: Performance of the protocol on different environmental conditions

statistics are in table 6.1. The results shown that the frequencies on modulation and the cut frequency of the filter are enough to give resilience to the protocol but, in the case of a jammer, the filter is useless, but it is true for all RF communication systems in the presence of the proper jammer.

6.3 Stealthiness experiments

The stealthiness is another characteristic useful on particular scenarios, for example those which can be applied on malware, the stealthiness is a characteristic associated with the volume level, as we presented in previous sections, different volume levels enable the communication at different distances but with larger distances the communication is more noticeable, in order to

Age range (years)	**Number of people**
20-30	3
31-40	3
41-50	2
61-71	2

Table 6.2: Age ranges for the stealthiness experiment

Volume level	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
People hearing something with ambient noise	0	0	0	3	8	10	10	10	10	10	10	10	10	10	10
People hearing something in complete silence	0	3	5	8	10	10	10	10	10	10	10	10	10	10	10

Table 6.3: Results of the stealthiness experiment

identify which volume can be used in scenarios where the stealthiness is important, the following experiment is proposed. We asked 10 people of different ages (ranged from 23 to 71 years old, the details are in figure 6.2) to say if they hear something when a frame is transmitted, the subjects were located at 0.5 m of distance, then the volume level is varied from 1 to 15 in each frame transmission and the experiment is repeated in two different conditions, in the presence of ambient noise (background music and others people speech in a crowded room) and in complete silence, these two approaches will give us an idea of the volume we can use to remain stealthy on transmission. The results are shown in table 6.3. From the results, we can see that levels under 4 can be used for stealthy communication, a very noisy environment could make possible the use of higher levels masking the communication taking as base the fact that the filter gives resilience to the ambient noise. Scenarios of complete silence should be avoided because even at lower levels the communication can be noticed despite their volume.

6.4 Comparison with the state of the art

As we saw in the previous experiments the proposed protocol has certain flexibility, it could be adapted for many purposes, we can vary all involved parameters to achieve the final goal of the client application who will make use of it. Similar works are focused on the feasibility of sound communication, in Hasan *et al.* [13] a communication with a PC is proposed. The

	Bitrate	**Type of communication**	**Type of proposal**
SoundComm-CISEG	490 bits/second	Mobile device to mobile device	Layer 1 and 2 protocols
Sensing-enabled Channels for Hard-to-detect Command and Control of Mobile Devices [13] (Direct channel)	1 bit/second	PC to mobile device	Proof of concept
Inaudible Sound as a Covert Channel in Mobile Devices [14]	345 bits/second	Mobile device to mobile device	Proof of concept

Table 6.4: Comparison of the present work with the state of the art

PC sends an acoustic signal to the phone at a rate of 1 bit/second, this is not directly comparable with our work, but it can be taken as reference. In Deshotels [14], a communication mobile to mobile device is proposed. This work has noticeable differences with the Deshotel's. The SoundComm-CISEG protocol has achieved a higher bit rate and it forms an entire stack of protocols who use the sound as a carrier signal while on Deshotels only a proof of concept is shown. The details of this comparison are shown in table 6.4.

In general, the present work proposes a functional stack of protocols that can be used by different applications which want to evade standard network controls using an alternative channel of transmission. Additionally, our tests show the flexibility of the protocol stack, and the capacity of adaptability for different usages.

6.5 Conclusions

In this work two protocols were presented for the physical and the data-link layer respectively, all the validation of the parameters was done in a testbed, this testbed was programmed at API level so the constraints of working inside a container are present, i.e., the access to the hardware and the memory are limited. Despite these constraints the proposal is capable to achieve higher bitrates than the state of the art and the communication can be unnoticed depending on the scenario and the environmental noise.

The chosen methodology was useful since the nature of the work is entirely practical. At the end of the research, the main contribution is a data-link protocol which uses the sound as a carrier signal for the communication between android devices which is capable of out-of-band transmission for evasion of network controls. The protocol has the desired characteristics, it operates in simplex mode, it is point to point,is based in the IEEE 802 standard and uses an alternative channel in order to avoid radio frequency transmissions consequently evading network controls, in this aspect, this work is effectively achieving the purpose of the research.

6.6 Further research

The present work can be improved in many aspects, first to improve the performance of the protocol, we can adapt the protocol to work at kernel level, since we have worked inside a dalvik virtual machine, several limitations are imposed, for example, the priority of the threads cannot be modified to fasten the process of demodulation, also the memory constraints inside the container of an Android application could be avoided working at lower levels.

The stealthiness can be also improved with a more complex modulation scheme , for example, a continuous phase modulation. As stated on Deshotels [14] the abrupt transition between modulation symbol's phase can produce audible clicks, and this can be avoided for example with continuous phase FSK (CPFSK).

Finally, the architecture of the protocol makes the system modular and capable of extension, for example to implement more complex flow controls, and enable half-duplex communication, and other communication system functionalities.

Bibliography

[1] Hartmut König. *Protocol Engineering*. Springer-Verlag Berlin Heidelberg, 2012.

[2] Ieee standard for local and metropolitan area networks: Overview and architecture. *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)*, pages 1–74, June 2014.

[3] Richard G. Lyons. *Understanding Digital Signal Processing (3rd Edition)*. Prentice Hall, 3 edition, November 2010.

[4] Hartmut König. Protocol engineering. pages 30–31. Springer-Verlag Berlin Heidelberg, 2012.

[5] Hartmut König. Protocol engineering. page 67. Springer-Verlag Berlin Heidelberg, 2012.

[6] Iso/ieee international standard - information processing systems local area networks - part 2: Logic link control. *ISO 8802-2 IEEE 802.2, First Edition 1989-12-31 (Revision of IEEE Std 802.2-1985)*, pages 1–114, Dec 1989.

[7] 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. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, March 2012.

[8] G. Suarez-Tangil, J.E. Tapiador, P. Peris-Lopez, and A. Ribagorda. Evolution, detection and analysis of malware for smart devices. *Communications Surveys Tutorials, IEEE*, 16(2):961–987, Second 2014.

[9] Claudio Marforio, Hubert Ritzdorf, Aurélien Francillon, and Srdjan Capkun. Analysis of the communication between colluding applications

on modern smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 51–60, New York, NY, USA, 2012. ACM.

[10] Ahmed Al-Haiqi, Mahamod Ismail, and Rosdiadee Nordin. A new sensors-based covert channel on android. *The Scientific World Journal*, 2014:14, 2014.

[11] J.-F. Lalande and S. Wendzel. Hiding privacy leaks in android applications using low-attention raising covert channels. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 701–710, Sept 2013.

[12] E. Fernandes, B. Crispo, and M. Conti. Fm 99.9, radio virus: Exploiting fm radio broadcasts for malware deployment. *Information Forensics and Security, IEEE Transactions on*, 8(6):1027–1037, June 2013.

[13] Ragib Hasan, Nitesh Saxena, Tzipora Haleviz, Shams Zawoad, and Dustin Rinehart. Sensing-enabled channels for hard-to-detect command and control of mobile devices. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, pages 469–480, New York, NY, USA, 2013. ACM.

[14] Luke Deshotels. Inaudible sound as a covert channel in mobile devices. In *8th USENIX Workshop on Offensive Technologies, WOOT '14, San Diego, CA, USA, August 19, 2014.*, 2014.

[15] Google. Android Compatibility Definition Documents, 2015.

[16] Google. Android documentation audiorecord, 2015.

[17] Google. Android documentation audiotrack, 2015.

[18] Scilab Enterprises. Scilab documentation noisegen, 2015.