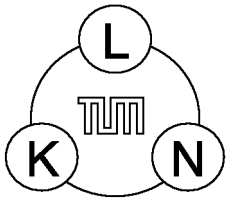

Technische Universität München
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. Jörg Eberspächer

Master's Thesis

Link-to-System Interfaces for System Level Simulations
Featuring Hybrid ARQ

Author:	Andrés Quiroga, Fernando
Matriculation Number:	3206385
Address:	Av. Jordán 30 08035, Barcelona Spain
Email Address:	fandres84@hotmail.com
Supervisor:	Jan Ellenbeck
Begin:	01. April 2008
End:	11. November 2008

ABSTRACT

Within the continuous evolution of wireless communications, new and ambitious requirements are planned to be met by next generation of mobile communications. In order to achieve those requirements, new technologies and mechanisms that work well over broadband frequency, like OFDM and OFDMA, need to be investigated, developed and tested in simulators.

New cellular systems designs are based on exploiting instantaneous channel conditions, improving the system performance. Due to this, system level simulations must support a Physical Layer (PHY) abstraction which accurately predicts the instantaneous performance of the link layer.

In order to accomplish this, a new link-to-system (L2S) interface has been developed and implemented in OpenWNS, a system level simulator for evaluation of OFDM systems developed at RWTH, Aachen. This interface is mainly realized through a set of mapping mechanisms, used to provide a BLER given a received coded block SINR.

Together with the theoretical study and evaluation of these mappings mechanisms, this thesis will analyze and evaluate Hybrid Automatic Repeat Request mechanisms (H-ARQ), and how these protocols can be implemented in system level simulators working together with the L2S interface.

This thesis shows how the new L2S interface is more accurate than past approaches, providing a gain of around 3 dB. Concerning H-ARQ protocols, the results show how the use of these new techniques provides a considerable gain with respect to normal ARQ or not using any such technique.

CONTENTS

ABSTRACT ..... **III**

CHAPTER 1 ..... **1**

INTRODUCTION 1

CHAPTER 2 ..... **3**

THE LINK-TO-SYSTEM INTERFACE 3
2.1. Introduction *3*
2.1.1 Link Level Simulations 3
2.1.2 System Level Simulations 4
2.1.3 Link to System Interface 5
2.2. Link-to-System Mapping Mechanisms *6*
2.2.1 Effective SINR mapping 7
2.2.1.1 Introduction 7
2.2.1.2 Types 8
2.2.1.3 Mutual Information Effective SINR Mapping 9
2.2.1.3.1 Received Bit Mutual Information Rate, RRBIR 9
2.2.1.3.2 Mean Mutual Information per Bit, MMIB 10
2.2.1.4 Exponential Effective SINR Mapping 10
2.2.2 BLER Mapping 11
2.2.2.1 Mathematical Alternative 11

CHAPTER 3 ..... **13**

HYBRID ARQ 13
3.1. Introduction *13*
3.1.1 Forward Error Correction 13
3.1.2 Automatic Repeat Request 13
3.2. Hybrid ARQ *15*
3.2.1 Introduction 15
3.2.2 Chase Combining 16
3.2.3 Incremental Redundancy 17
3.2.3.1 Puncturing and Signaling 18
3.2.3.2 Different Proposals 19
3.2.4 Type 3 H-ARQ 20
3.3. Different H-ARQ Schemes Comparison *21*
3.4. History *21*
3.4.1 HSPA 21
3.4.1.1 HSDPA 21
3.4.1.2 HSUPA 22
3.4.2 LTE 23

CHAPTER 4 ..... **24**

LINK-TO-SYSTEM INTERFACE IN OPENWNS 24
4.1. Introduction *24*
4.1.1 Effective SINR mapping 24
4.1.1.1 MIESM theoretical computation 25
4.2. Link-to-System Mappings Implementation *26*
4.2.1 Tables Available 27
4.3. Link-to-System Mappings Evaluation *28*
4.3.1 Effective SINR Mapping 28
4.3.1.1 Origin of the Tables 28
4.3.1.2 Influence of the Parameters in MIESM 29
4.3.1.2.1 SINR 29
4.3.1.2.2 Modulation 30
4.3.2 Mutual Information to BLER Mapping 31

4.3.2.1 Origin of the Tables 31
4.3.2.2 Influence of the parameters in MI to BLER mapping 32
4.3.2.2.1 Mutual Information per Bit 32
4.3.2.2.2 Coding Scheme 33
4.3.3 ESM and MI to BLER mapping together 37
4.4. OpenWNS versus WiMAC L2S Interface 38

CHAPTER 5 42

EVALUATION OF HYBRID ARQ SCHEMES 42
5.1. LTE Physical Layer 42
5.2. Evaluation of H-ARQ Protocols 43
5.2.1 Chase Combining 44
5.2.1.1 PHY Abstraction for Chase Combining 44
5.2.1.2 Simulation Scenario for Chase Combining 44
5.2.1.3 Performance Evaluation 45
5.2.1.3.1 BLER 45
5.2.1.3.2 Throughput 46
5.2.1.3.3 Modified Scenario 47
5.2.2 Incremental Redundancy 49
5.2.2.1 PHY Abstraction 49
5.2.2.1.1 No Repeated Coded Bits Allowed Scenario 49
5.2.2.1.2 Repeated Coded Bits Allowed Scenario 50
5.2.2.1.3 Intermediate Scenario 51
5.2.2.2 Simulation Scenario 51
5.2.2.3 Performance Evaluation 53
5.2.2.3.1 BLER 53
5.2.2.3.2 Throughput 56
5.2.3 H-ARQ Schemes Comparison 56
5.3. Advices to Implement H-ARQ in OpenWNS 61

CHAPTER 6 62

CONCLUSIONS 62

APPENDIX A 64

RBIR CALCULATION 64

APPENDIX B 69

SOFT COMBINING 69

APPENDIX C 71

OPENWNS 71

APPENDIX D 73

TURBO CODES 73

LIST OF FIGURES 75

LIST OF TABLES 77

GLOSSARY 78

BIBLIOGRAPHY 79

CHAPTER 1

Introduction

Cellular communications are continuously evolving. Even though the current generation of mobile communications is supposed to remain being the technology used for some more years, new technologies, like Long Term Evolution (LTE) or Worldwide Interoperability for Microwave Access (WiMAX), are currently being studied and tested.
Goals of these new technologies include improving spectral efficiency, lowering costs, improving services, making use of new spectrum and better integration with other open standards.

Some of those goals are in part achieved by exploiting the channel instantaneous conditions. In order to be able to evaluate this new scenario, current system level simulations must support a Physical Layer (PHY) abstraction which accurately predicts the instantaneous performance of the PHY link layer. To deal with this new PHY abstraction within system level simulations, a new Link-to-System (L2S) interface has been developed. This interface is mainly realized through a set of mapping mechanisms.

Another technique that will help new systems to achieve the goals mentioned above is Hybrid ARQ protocols (H-ARQ), already implemented for High Speed Packet Access (HSPA).

This thesis provides a theoretical study regarding these two fields, the new L2S interface, focusing on its mapping mechanisms, and H-ARQ protocols. After that, a performance evaluation in both fields is given. Simulations are run using OpenWNS, an open source simulation platform for wireless and multi-cellular mobile communications develop at RWTH, Aachen.

Further descriptions of the concepts introduced so far can be found along the thesis, but it seems interesting to introduce some general ideas.

A link level simulator implements the main blocks of the communication chain, and models the wireless channel. Due to the computational cost of running those simulations when higher layers performance wants to be evaluated, link level simulations are typically performed in advance, and the results obtained are stored. Those results are used then to easily model the PHY layer behaviour in system level simulations, which deal with higher layer events.

The L2S interface is considered the set of functions used by system level simulators to introduce and use results from link level simulations within their implementation.

Within the L2S interface, some mapping mechanisms are used in order to provide a Block Error Rate (BLER) given a received coded block Signal to Interference plus

Noise Ratio (SINR). This thesis analyzes among other concepts why these mechanisms are needed, how they work, how they have been implemented in system levels simulators like OpenWNS and which results they provide.

The second field of study within this thesis is H-ARQ protocols, a new generation of ARQ protocols. H-ARQ improves the system throughput and provides robustness against occasional transmission errors by combining, rather than discarding, failed transmission attempts with the current attempt, effectively creating a more powerful code.

This thesis will present a study on the history of those H-ARQ protocols, the different existing approaches, how they work, and finally and very important, how they can be implemented in OpenWNS working together with the L2S mapping mechanisms and which results they would provide.

The thesis is organized as follows.

CHAPTER 2 provides a theoretical view of system level and link level simulations and the L2S interface, with special attention to the mapping mechanisms used to provide a coded BLER from a received coded SINR.

The second field of study within this thesis, H-ARQ protocols, is analyzed in CHAPTER 13. The study includes some background, an analysis of the different existing types of H-ARQ and a future overview.

CHAPTER 14 presents a performance evaluation of the L2S mapping mechanisms implemented in OpenWNS. Some more information about the simulator can be found in Appendix C.

CHAPTER 5 includes a more practical study of H-ARQ protocols. A series of tests have been implemented in order to predict the performance obtained by the use of such techniques. A comparison between different schemes is given, as well as some advices to take into account when implementing H-ARQ in system level simulators in general and OpenWNS in particular.

Finally, the conclusions of this thesis are included in CHAPTER 6.

CHAPTER 2

The Link-to-System Interface

2.1. Introduction

As it was briefly presented in the introduction, system level simulations use results from link level simulations. The mechanisms used for system level simulators to introduce those results within their implementation compose what it is called the L2S interface. This interface is mainly realized through a set of mapping mechanisms.

Those mapping mechanisms constitute one of the fields of study within this thesis, and they will be further analyzed along the text. But before a more detailed description about them, it seems important to clarify their existence's motivation.

In order to do that, it would be interesting to go a little bit backwards and first explain why general link level and system level simulations are performed.

2.1.1 Link Level Simulations

Link layer simulators model the PHY and the channel behaviour. Starting from the transmitter components and ending with the receiver components, everything must be modelled in the more accurate but still computationally admissible possible way.

Figure 1 shows an example of a digital communication chain, including the main steps that take place during the process.

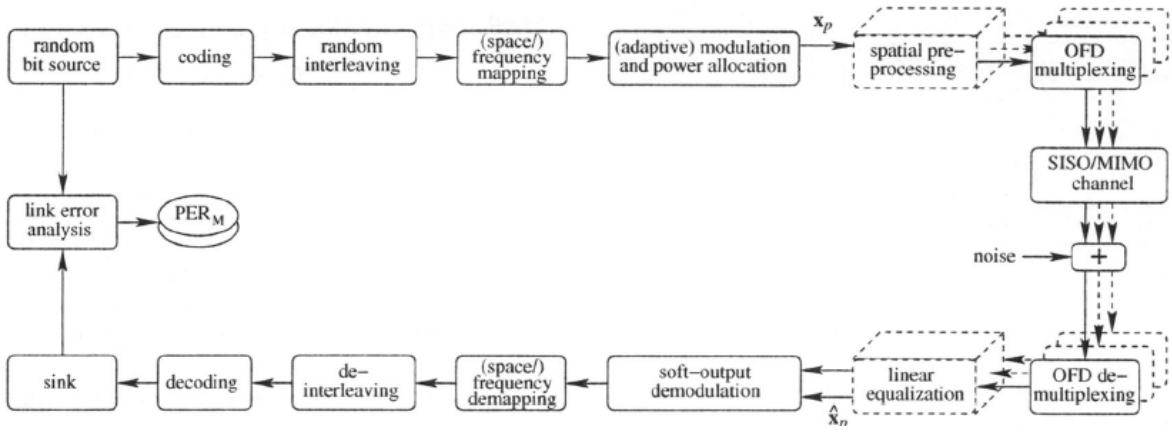

Figure 1 *Example of a digital communication chain [1]*

Everything starts with a bit source. Those bits are the payload, the information that wants to be transmitted. They can be just randomly created if only the communication chain want to be modelled or they can come from higher levels simulations, if a more general overview of the system wants to be performed.

The bits to be transmitted are coded in order to protect the information. That means some redundant bits are added to the original ones. The coding process has been widely studied during the last decades, and there are a lot of schemes available in the literature.

Lately, turbo coding is being used, due to its ability to approach the Shannon limit [2], the theoretical limit of maximum information transfer rate over a noisy channel. Turbo codes make it possible to increase data rate without increasing the power of a transmission, or they can be used to decrease the amount of power used to transmit at a certain data rate.

After the coding, actions directed to protect the information and improve the packet chances of good reception can be performed, such as the random interleaving. The random interleaving consist of modifying the position of the bits belonging to a set of packets following a concrete pattern, avoiding the concentration of errors in bits of the same original block.

When working with Multiple Inputs Multiple Outputs systems, MIMO, techniques like space or frequency mapping can be used to increase the SINR at the receiver, to combat the received signal's fading and to reduce the interferences or to increase the system capacity. Further details can be seen at [3].

Then, the information is modulated and sent through the channel. During its travel, information can be partially corrupted due to channel conditions, like shadowing or fast fading, dispersive aspects like multi path propagation and inter symbol interference, and due to physical propagation laws, like the path loss.

Finally, the data arrives at the receiver, which can measure the received coded bits SINR. The receiver performs the reverse actions done at the transmitter, and tries to recover the original packet. A BLER can be then obtained, and it acts sometimes as a performance indicator of how good the transmission was and how good is the receiver.

That would be a simple and non detailed explanation about what happens in a real transmission. Link level simulators try to model this behaviour in a computationally efficient way.

As it was outlined in the introduction, performing link level simulations has a high computational cost. Due to this, these simulations are normally performed in advance, and the results obtained are stored. Then, those results can be easily used to model the PHY behaviour when other higher level issues want to be evaluated, avoiding lots of calculations.

2.1.2 System Level Simulations

System level simulations deal with higher level events or issues, like interference management, resource allocation management, power allocation, etc. When those

issues want to be evaluated, the PHY behaviour is modelled by using the stored results from link level simulations performed in advance.

2.1.3 Link to System Interface

Once link level and system level simulations have been briefly presented, it is time to detail the L2S interface.

Figure 2 exemplifies the link performance model, which would represent the process that a L2S level simulator follows in order to provide the desired BLER, commonly used to finally decide whether the transmission has been successful or not, taking the suitable decisions in each case after that.

The locations of link level and system level simulations, as well as the L2S interface are depicted in the figure.

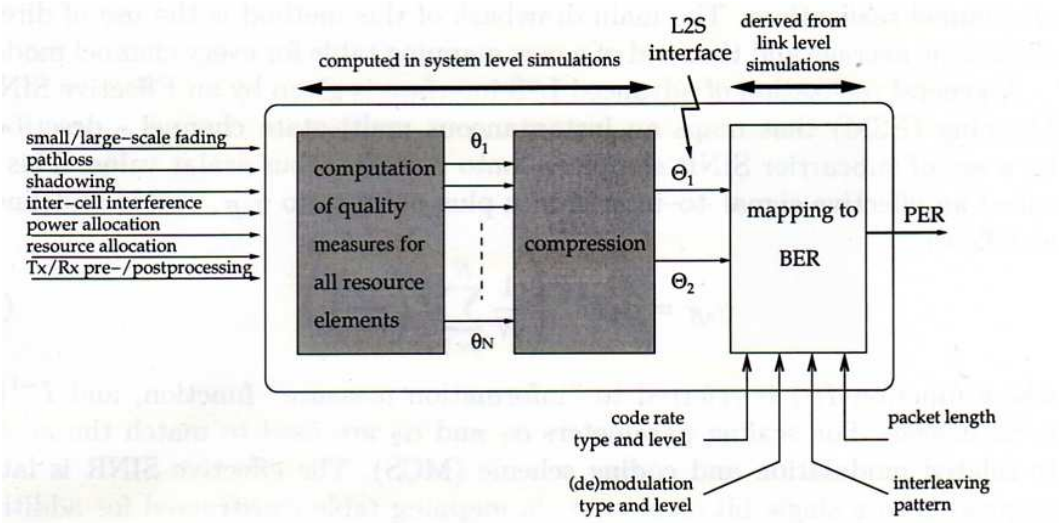

Figure 2 *Link performance model [4]*

It can be seen that a lot of parameters feed the model. Some of them are set by the researcher depending on his needs, and some others depend on the simulator implementation itself. Some of them represent channel conditions, some power and resource allocation, and some modulation coding scheme and block length characteristics.

The number of parameters is typically too large to be directly used to obtain a BLER. Moreover, each one of them comes or represents a completely different physic behaviour or configuration parameter. Therefore, those parameters on the left side of the figure are map into a set of quality measures, which somehow gather all the information provided by the parameters. These quality measures are normally SINRs, due to it has been proved that SINRs accurately describe all the information provided by the parameters. However, a new compression is normally desired and performed, to facilitate the last mappings and finally provide a BLER. Due to this compression, just one SINR is finally used to obtain the BLER of a specific channel or user's state.

There it is located one of the mapping mechanisms that this thesis will further analyze. It maps a set of quality measures, SINRs, into a single scalar value, SINR.

Once this single scalar value is obtained, a final mapping is performed, and a BLER value is achieved. Some parameters determine the result of this mapping. They can be seen at the bottom right side of Figure 2 . This is the second mapping mechanisms that will be further studied within this thesis. It takes a single scalar parameter which represents the channel state, SINR, and it provides a BLER value. Further details can be seen in the next sections.

It is interesting to see, as it is explained in the figure itself and it was already mentioned, that this last map uses values derived from link level simulations performed in advance, and whose results were stored.

Summarizing, two mapping mechanisms located into the L2S interface have been presented.

2.2. Link-to-System Mapping Mechanisms

Up to now, link level and system level simulations have been briefly presented. In the middle of these two, acting as a kind of a linker, the L2S interface. It has been mentioned that the L2S interface is mainly realized through a set of mapping mechanisms, specifically two, used to obtain the BLER of a coded block.

The first map goes from a set of instantaneous channel state measures, SINRs, and provides a single scalar value, SINR. The second one, maps from this SINR into a BLER value. Figure 2 can be seen now as Figure 3 , where the quality measures are now SINRs.

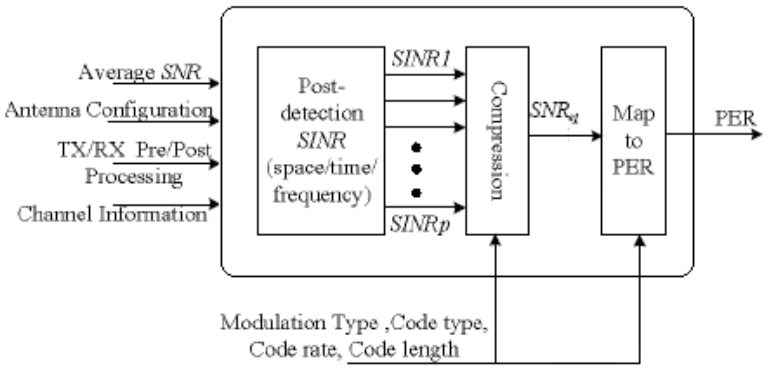

Figure 3 *Link Layer abstraction procedure [5]*

A vector of SINR comes out from the first computation, and it is compressed into a single value, normally called effective SINR, $SINR_{eff}$ in the figure. This first mapping is then well known as Effective SINR Mapping.

The effective SINR value is then directly mapped into a BLER or PER value. This mapping does not have a well known name. In that thesis, even though up to now it does not make sense, it will be called Mutual Information to BLER mapping. Both mapping will be further studied from now on.

2.2.1 Effective SINR mapping

2.2.1.1 Introduction

As it was explained, this mapping compresses a vector of SINRs into a single SINR value, called effective SINR.

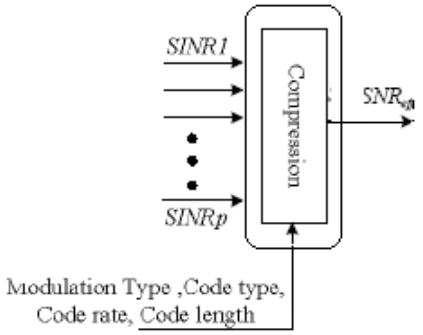

Figure 4 *Effective SINR Mapping*

Let's analyze in detail this mapping.

The meaning of the vector of SINRs, and the desire of compress this values into a single effective SINR, come from the future use of OFDM and OFDMA within LTE and WiMAX. Within OFDM, a large number of closely-spaced orthogonal sub-carriers are used to carry data. The user data are divided into several parallel data streams or channels, one for each sub-carrier.

OFDM systems might experience selective fading, and each one of the sub-carriers may be received with a different channel gain. Therefore, the result of a transmissions of a large encoded packet is encoded symbols with unequal SINRs. Thus, in order to evaluate the user's general performance or the general channel state it becomes necessary to deal with a vector of SINRs, like the one shown in Figure 4 . The effective SINR value is desired then because the link level curves which will facilitate the mapping into a BLER are generated assuming frequency flat channel response at given SINR, thus they need one single SINR value.

The first idea then would be to directly average those SINRs values and obtain the effective SINR value.

$$SINR_{eff} = \frac{1}{N}\sum_{n=0}^{N-1} SINR_n \qquad (2.1)$$

This formula represents the simple direct average of a set of SINR values, giving the same importance to each of them. It does not present any mathematically problem. However, it should not be used to calculate the effective SINR. There are some reasons that explain this, and they will be analyzed from now on.

First, because the information sent through each one of the different OFDM sub-carriers might have been modulated differently from the rest. The modulation scheme

determines the symbol constellation, thus the distance between consecutive symbols. Therefore, the difficulty to distinguish between symbols within a constellation at a given SINR is not the same, and it is easier to distinguish symbols in a BPSK modulation than in a 64QAM. Consequently, the modulation scheme used for each one of the sub-carriers must be taking into account within the averaging, in order not to lose this important information.

Second, due to the possible existence of aberrant values. For example, if a user data has been distributed along 8 sub-carriers, and 7 of them have been received with poor SINR, and just one with a great SINR, it would be possible that a direct average determines that the whole transmission has been successful, when in fact, most of the information could not be decoded.

Summarizing, some techniques that somehow fairly average the vector of SINRs, reducing the impact of aberrant values and taking into account the modulation scheme used, were required. Those techniques are actually called Effective SINR Mappings.

2.2.1.2 Types

Once it was clear the need of developing some techniques to fairly average the different SINR resultant of an ODFM transmission, taking into account the modulation scheme, different proposals were presented.

Those proposals which remained can be found in the literature. In general, they all can de described as follows [6]:

$$SINR_{eff} = \phi^{-1}\left(\frac{1}{N}\sum_{n=0}^{N-1}\phi\left(SINR_n\right)\right) \tag{2.2}$$

Where $SINR_n$ is the SINR in the n^{th} sub-carrier, N is the number SINRs to average, and ϕ is an invertible function which depends on the modulation coding scheme, as it will be further analyzed in CHAPTER 1.

Function ϕ determines the difference between the diverse approaches. It changes the SINR domain into another in order to work with values that can be fairly averaged.

The most well known approaches are:

i) **Exponential Effective SINR Mapping, EESM**, where ϕ is derived from the Chernoff bound on the probability of error [7].

ii) **Mutual Information Effective SINR Mapping, MIESM,** where ϕ is derived from the constrained capacity. Modulation constrained capacity is the mutual information of a symbol channel [6].

The problem presented then is solved with the use of these techniques, and now the effective SINR obtained represents in a more accurately way the information provided by the vector of SINRs.

2.2.1.3 Mutual Information Effective SINR Mapping

Mutual information effective SINR mapping uses nonlinear mapping relationships to calculate a mutual information value given a SINR.

Before further considerations about MIESM, it would be interesting to remind what mutual information is.

In probability theory and information theory, the mutual information of two random variables is a quantity that measures the mutual dependence of the two variables [8]. Formally, the mutual information of two discrete random variables X and Y can be defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p \sum_{i=1}^{n} X_i(x,y) \log\left(\frac{p(x,y)}{p_1(x)\, p_2(y)} \right) \tag{2.3}$$

Where $p(x,y)$ is the joint probability distribution function of X and Y, $p_1(x)$ and $p_2(y)$ are the marginal probability distribution functions of X and Y respectively. The marginal distribution of a set like X provides the probabilities of each one of the possible values within this set, x_i. I is the mutual information.

The MI concept used in OpenWNS differs a bit from the theoretical one presented above. It will be explained in 4.1.1.1.

Taken up again MIESM concept, there are two methods to obtain the MI per coded bit (MIB).

The key formula used [6] in both cases is:

$$SINR_{eff} = \alpha_1 . M^{-1}\left(\frac{1}{N} \sum_{n=1}^{N} M\left(\frac{SINR_n}{\alpha_2} \right) \right) \tag{2.4}$$

Where N is the number of SINRs to average, M is an invertible function that determines the MI and α_1, α_2 adjusts the method for a specific modulation scheme.

2.2.1.3.1 Received Bit Mutual Information Rate, RRBIR

This method derives the MI per coded bit from the received symbol MI that comes from the system level.

Depending on the receviers used, Linear or Maximum-Likelihood, and the scheme of antennas, SISO or MIMO, there are different approaches to calculate the desired MI per coded bit [6]. Further details about how to calculate the MI can be found in Appendix A.

This method is the one that is used within OpenWNS to calculate the MI.

2.2.1.3.2 Mean Mutual Information per Bit, MMIB

This alternative method derives the MI per coded bit directly from the bit channel itself that comes from the system level. Further details can be seen in [6].

As it happened for the RRBIR approach, different formulas are used depending on the receivers and the scheme of antennas used.

2.2.1.4 Exponential Effective SINR Mapping

The key formula used in EESM is [6]:

$$SINR_{eff} = -\beta \ln\left(\frac{1}{N}\sum_{n=1}^{N}\exp\left(-\frac{SINR_n}{\beta}\right)\right) \quad (2.5)$$

Where N is the number of sub-carriers, thus SINRs, used in an OFDM system, and β is adjusted to match the effective SINRs to a specific modulation scheme. A suitable β value for each modulation of interest can be found through adequate link-level simulations. Some interesting results are shown in Table 1, from [7].

Table 1 β values obtained after simulation assuming channel model ITU pedestrian 1, and channel model ITU pedestrian 2.

MCS	ITU pedestrian 1 β	ITU pedestrian 2 β
4QAM CR 1/2	1.59	1.67
4QAM CR 3/4	1.7	1.75
16QAM CR 1/2	5.33	5.41
16QAM CR 3/4	8.45	7.94
64QAM CR 1/2	16.10	17.27
64QAM CR 3/4	27.21	32.33

EESM has been shown to provide accurate results for calculating BLER [7]. Therefore, it can be used to predict BLER, both for simulation and as a tool for the link adaptation (L2S mapping techniques).

Additional characteristic of EESM would be the following. From (2.5), and remembering the e^{-x} shape, it can be appreciated that EESM puts higher weight on lower SINR samples than in higher SINR samples. Thus, the critical lower SINR samples dominate the effective SINR average. That makes this effective mapping approach to be considered as a conservative one.

2.2.2 BLER Mapping

The BLER mapping goes from an effective SINR to a BLER. Again, the MCS influences the result, as well as the coded Block Length (BL).

It will be further explained later, but is now important to remark that OpenWNS does not calculate the effective SINR value. Instead, it remains in the mutual information domain, keeping the MI values in order to finally provide a BLER value. Due to this, this last mapping will be called MI to BLER mapping from now on.

Figure 5 represents the abstraction proceeding of the BLER mapping, assuming an effective SINR is calculated (left side of the figure), and the proceeding assuming the mapping goes directly from a MI value into a BLER value (right side), like in OpenWNS.

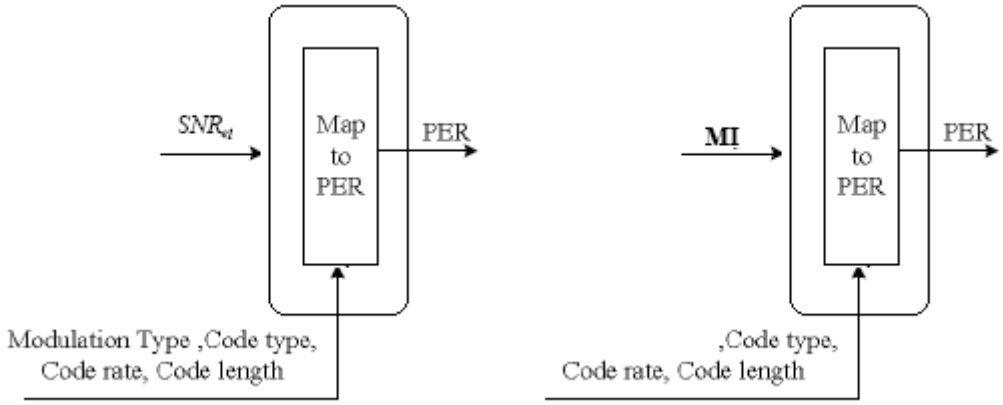

Figure 5 *BLER mapping, assuming it provides a BLER from a SINR value or MI value [5]*

Normally, this mapping is performed by using look-up tables, where results from sophisticated link level simulations are stored, relating SINR/MI and BLER values. Those tables will be further studied within the practical part of this thesis.

2.2.2.1 Mathematical Alternative

There is a mathematical alternative to perform the SINR to BLER mapping, presented in [6]. The study proposes a parametric function to approximate the curve obtained with the link-level simulations, considering a Gaussian cumulative model.

After some parameterization, the BLER for each MCS depends on two parameters, related with the Gaussian distribution model used.

$$BLER_{MCS} = \frac{1}{2}\left[1 - erf\left(\frac{x - b_{MCS}}{\sqrt{2}c_{MCS}}\right)\right], c \neq 0 \qquad (2.6)$$

Where x is the SINR value. The actual dependency is further analyzed in [6].

It has been shown that the mapping from MIB to BLER can be assumed independent from the modulation alphabet. Thus, the formula above can be seen from now on as

$$BLER_{CS,BL} = \frac{1}{2}\left[1 - erf\left(\frac{x - b_{CS,BL}}{\sqrt{2}c_{CS,BL}}\right)\right], c \neq 0 \qquad (2.7)$$

Where b and c depend on the coding scheme and the block length used, but not on the modulation used anymore.

This model is proposed due to the particular physical interpretation of b and c. b is closely related to the code rate, while c is also related to the block size. Some values for b and c for different CS and BL can be seen at [6].

By using this parametric function, everyone can obtain the SINR to BLER tables for different CS just knowing the suitable values for b and c.

CHAPTER 3

Hybrid ARQ

3.1. Introduction

Hybrid ARQ (H-ARQ) is essentially a combination of Forward Error Correction (FEC) and ARQ in an optimal manner. H-ARQ improves the system throughput and provides robustness against occasional transmission errors by combining, rather than discarding, failed transmission attempts with the current attempt, effectively creating a more powerful code.

Before focusing on H-ARQ, a brief introduction of FEC and simple ARQ is included. Both techniques are used due to the characteristics of wireless channels, where a series of obstacles and physics characteristics can damage the information transmitted.

3.1.1 Forward Error Correction

FEC is an error control system for data transmission. The system adds redundant bits to the original message, fortifying the data and allowing the receiver to detect and correct errors without requiring new retransmissions.

The main advantage of using FEC is that no reverse channel is needed. Contrary, the code rate decreases. Therefore, FEC by itself is normally used when the cost of having a back channel or losing throughput because of retransmissions is higher than using a lower code rate.

3.1.2 Automatic Repeat Request

ARQ is an error control method for data transmission too. It uses acknowledgments (ACK) and timeouts to achieve reliable data transmission.

An ACK is a message that the receiver sends to the transmitter when the packet has been successfully received. The timeout is the maximum time that the transmitter waits for an ACK after having sent the packet before consider it lost.

There are different well known ARQ techniques. The use of one or another is defined depending on the kind of data transmitted, the maximum delay supported or the possibility to have buffers in both the transmitter and the receiver.

A brief description of these methods:

i) Stop and Wait:

Only one packet can be on the channel at the same time. The transmitter has a one packet size buffer. There is no buffer in the receiver. The ACKs and NACKs are not numbered.

Note that the non reception of ACK in a time interval predefined is considered like a NACK.

ii) Go Back N:

Up to N packets can be travelling though the channel simultaneously. The transmitter has an N packet buffer. There is no buffer in the receiver, so all the packets previously sent and not ACKed must be retransmitted after a NACKed packet with higher sequence number. ACKs and NACKs are numbered, meaning that they refer to a specific packet.

iii) Selective repeat:

There are buffers in both transmitter and receiver, so only the erroneous packets are transmitted again.

ACKs and NACKs are numbered.

There must be a window defined, allowing the transmission of a maximum number of packets not ACKed.

iv) N channel Stop and Wait:

Each channel works like a simple Stop and Wait channel.

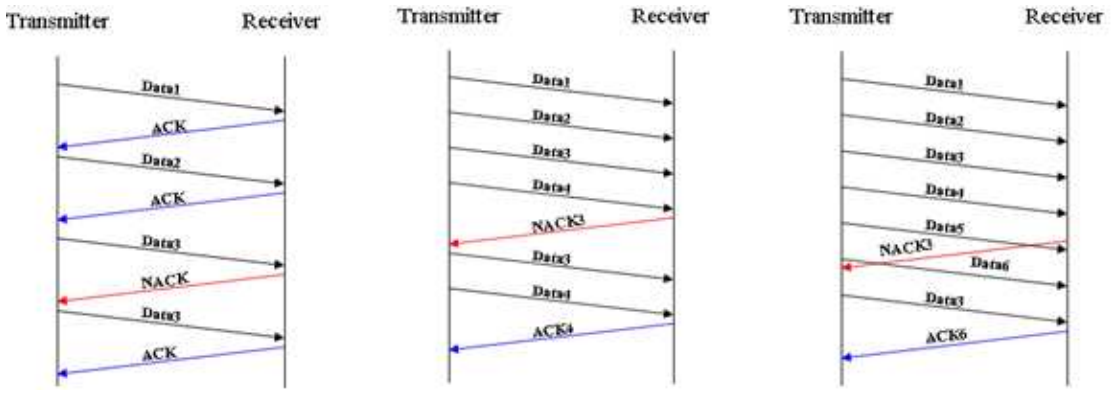

Figure 6 *Example of ARQ protocols. From left to righ: Stop and Wait, Go Back N, Selective Repeat*

3.2. Hybrid ARQ

3.2.1 Introduction

The goal of H-ARQ is to improve the performance of the system, by providing robustness against occasional transmission errors, handling variations in the instantaneous radio link and increasing the system capacity. H-ARQ protocols achieve these goals by combining the information provided by both previous erroneous transmissions and last attempt.

The concept is simple. Even though a packet has been damaged by the channel, it does not mean it can not still provide useful information. Up to the use of H-ARQ, this information was dropped. From now on, with the use of H-ARQ protocols, the damaged packet is stored in a buffer, at the same time a retransmission is required. Once the retransmission arrives to the receiver, both packets are combined, obtaining a single combined packet which is more reliable than its constituents. Details about how the bits from different transmissions are combined can be found in Appendix B. Figure 7 depicts the process.

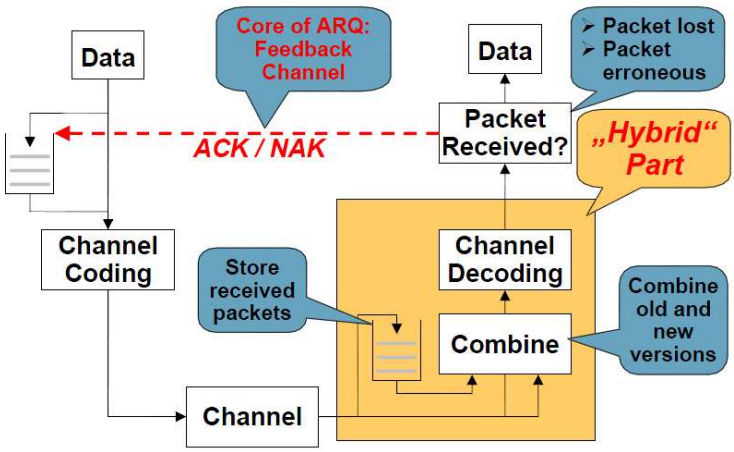

Figure 7 *H-ARQ scheme protocol [10]*

Once a packet is received, the protocol finds out whether the packet has been sent for the first time or not. In case there are stored previous transmissions of the same packets, they are combined. The resultant packet is sent to the decoder, which tries to recover the information. In case the packet arrives for the first time, it is directly sent to the decoder. Parallel, a copy of the packets is stored in the buffer. That copy will be erased once the decoding process of the packet results successfully.

The receiver will decide whether the transmissions has been successful or not, and it will inform the transmitter by sending an ACK or a NACK. If the information coded into the packet can not be recovered, the protocol will ask for a retransmission. A more detailed description of the protocol can be found in 3.4 and 5.3.

An H-ARQ retransmission has to represent the same set of information bits as the original transmission, but the bits chosen to represent this information in each retransmission may not be the same. Depending on whether the retransmission is

required to be identical to the original or not, H-ARQ protocols can be divided into different types [11].

There are three well known H-ARQ types.

i) **Type I, called Chase Combining (CC)**

ii) **Type II, Incremental Redundancy (IR)**

iii) **Type III.**

3.2.2 Chase Combining

The retransmissions consist of the same set of coded bits as the original transmission.

After each retransmission, the receiver combines each received bit with any previous transmissions of the same bit, and the combined signal is sent to the decoder.

Since each retransmission is an identical copy of the original transmission, Chase Combining does not give any additional coding gain but only increases the accumulated received Eb/No after each retransmission. The code rate remains always the same.

Figure 8 exemplifies the process, starting from the original data. Redundant bits are added to fortify the code. The same packet is sent every time a retransmission is required. It can be seen how the Eb/No and the CR evolves after each combination.

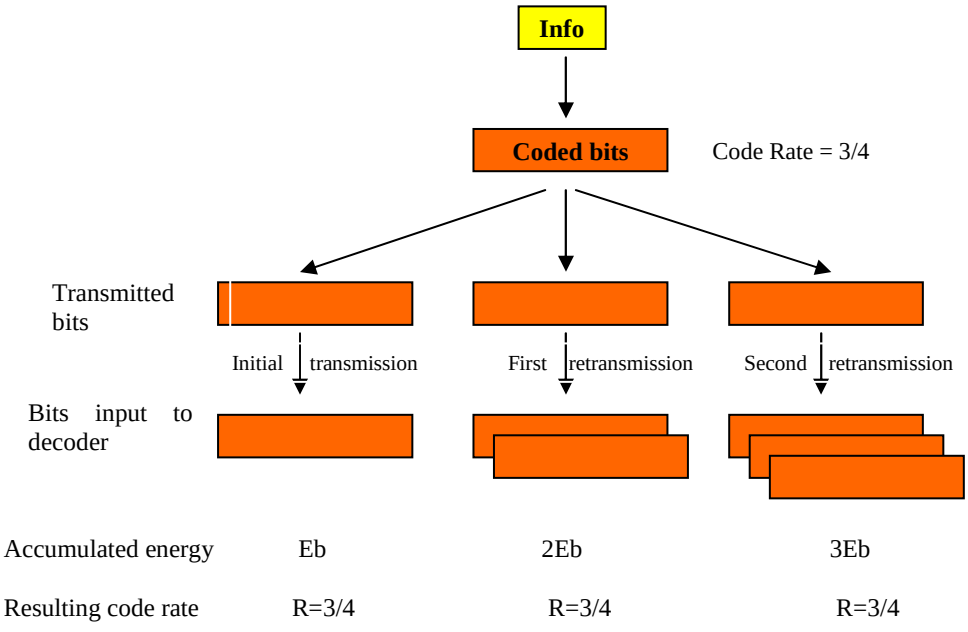

Figure 8 *Example Chase Combining H-ARQ [12]*

Advantages of Chase Combining compared to other H-ARQ methods (which will be studied after CC) include smaller decoder complexity, smaller memory requirements and the ability to self-decode every block before joint decoding.

3.2.3 Incremental Redundancy

Differently from Chase Combining, in an Incremental Redundancy scheme each retransmission does not have to be identical to the original transmission. Instead, multiple sets of coded bits are generated from the codeword. As it was commented, those bits must represent the same set of information bits.

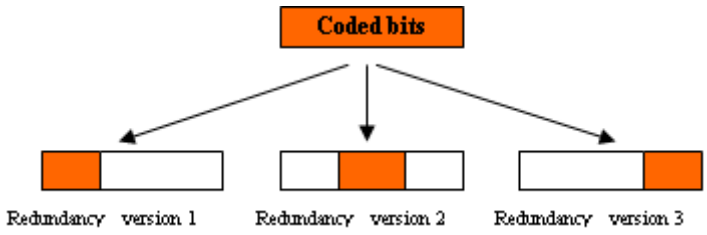

Figure 9 *Example of redundancy versions created from mother code*

Figure 9 exemplifies the process, where three redundancy versions, normally different, are created by puncturing the codeword.

Like it was done in CC, the receiver combines the retransmission with previous transmission attempts of the same packet.

Since new redundant bits not included in previous transmissions can be included into the last attempt, the resulting code rate is generally lowered by a retransmission. The energy per bit only increases in case some bits are transmitted more than once.

There are various techniques to decide which and how many bits will be included in each retransmission [12]. A first division between those techniques would be the following.

On one hand, those H-ARQ IR schemes which do not care about link adaptation. These H-ARQ protocols rely on the scheduler to correctly decide the code rate that should be used for every transmission depending on the channel prediction. Therefore, the H-ARQ process just has to worry about which bits send to the scheduler for the next attempt in case a retransmission is required.

On the other hand, those which introduces a channel sensing functionality, and adapt the code rate of the next retransmission based on the predicted SINR for the next attempt or the difficulty found to decode the previous packet.

Independently of which of the methods explained above is used, the combination of bits that will compose the new set of bits to be transmitted depends on the algorithm chosen too.

Typically, Incremental Redundancy is based on a low-rate code and the different redundancy versions are generated by puncturing the output of the encoder. The puncturing process consists of choosing a series of bits from the original packet following a concrete pattern. It is used to try to send the same information but using a higher code rate. Of course, the risk of not being able to finally decode the information increases.

In the first transmission only a limited number of the coded bits are transmitted, effectively leading to a high code rate. In the retransmissions, additional coded bits are transmitted, thus the resultant code rate decreases, as it was already mentioned, providing a coding gain. Figure 10 exemplifies the whole Incremental Redundancy scheme process.

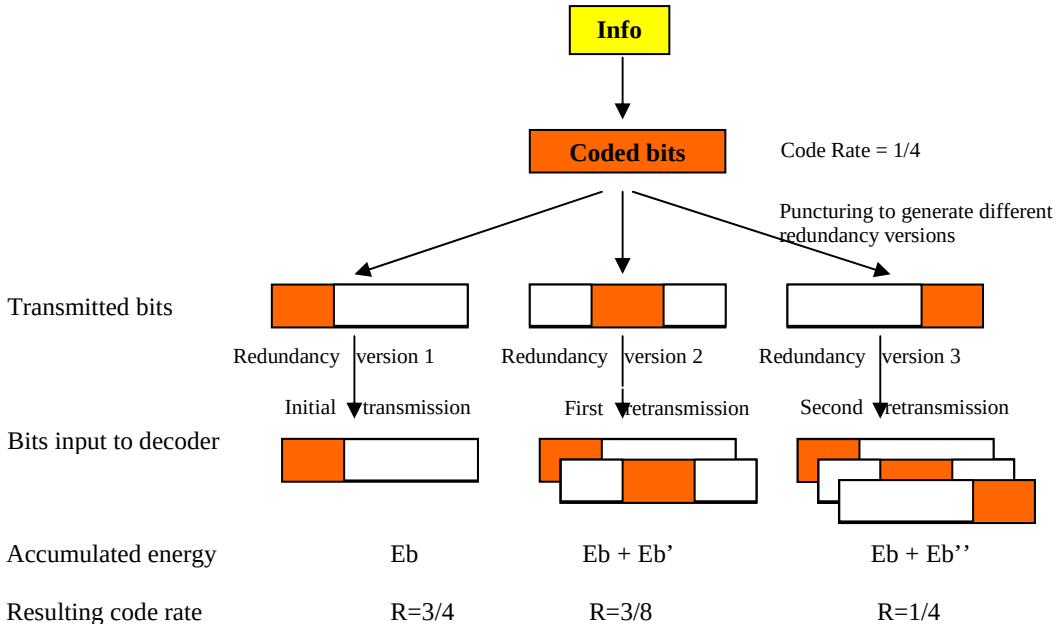

Figure 10 *Example Incremental Redundancy H-ARQ process [12]*

Where Eb' represents the energy per bit accumulated in case repeated bits are sent within the retransmissions.

3.2.3.1 Puncturing and Signaling

It has been commented that, typically, the different redundancy versions are generated by puncturing the output of the encoder. The puncturing generation process, however, is not random, and needs to satisfy a series of premises [12].

First, the set of bits included in the first transmission should provide good performance not only when used alone, but when used in combination with the code for the second transmission too. Same requirement apply for subsequent transmissions.

That means that the puncturing pattern must provide sets of bits that, given a code rate achieved after combining, provide similar performance than if these sets of bits would have been directly created thinking in use this code rate for the first time.

The second premise concerns the coding scheme used. As it was explained, the different set of bits must represent the same set of information bits. Therefore, it seems that each transmission/retransmission is equally important. However, this is not always true, and it depends on the coding scheme used. For example, with Turbo Codes, the systematic bits are more important than redundant bits. Consequently, it is important to assure that systematic bits are received correctly as soon as possible.

In order to satisfy that, systematic bits will be normally included into the initial transmission. However, this is not enough. If the first transmission was received strongly damaged, it would not matter how many more redundant bits were sent in consequent retransmissions, the information could not be recovered. To solve this, some sort of signalling is required. The receiver will specify if he needs new redundant bits or systematic bits into the next transmission.

3.2.3.2 Different Proposals

Even though the IR scheme explained up to now is the most typical algorithm, there are other IR techniques presented by important research projects that can be interesting to introduce.

The WINNER project proposes an algorithm called cyclic shift incremental code. It works as follows [13]:

The method starts from a mother code, the combination of systematic bits (original information) and parity bits (added to protect the information bits). Figure 11 represents a code word for a 1/3 code rate that will be used as an example.

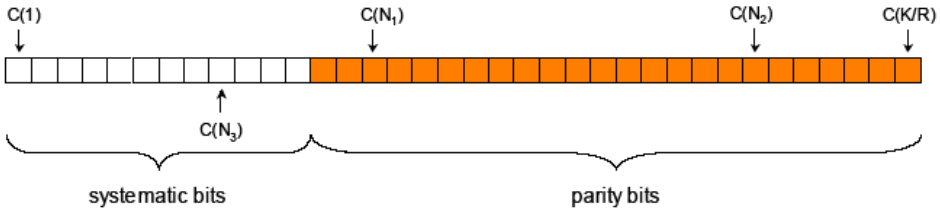

Figure 11 *Example Codeword with CR=1/3 [13]*

Before each retransmission, the codeword is stored in a buffer. Then, a number of coded bits will be selected depending on the modulation and coding scheme used. The first N1 bits are transmitted. If an error occurs and the receiver asks for a retransmission, the mechanism will select the next bits in a sequential order.

The size of the retransmission packet can changed following the desires of the receiver. Thanks to some signalling, the receiver can decide whether he prefers the same or less redundant bits as in the initial transmissions, with the aim of achieving throughput maximization.

If a new retransmission is required and there are no enough parity bits to full the data packet, some bits already transmitted will be part of the new transmission, again following a sequential order, and so forth.

Figure 12 exemplifies this process. Note that for the figure, the packet size is fixed for every transmission.

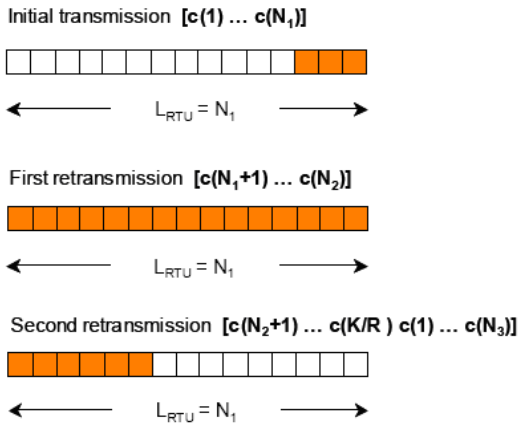

Figure 12 *Example of IR scheme proposed in WINNER Project [13]*

3.2.4 Type 3 H-ARQ

This scheme is similar to IR, and uses puncturing techniques to create the set of bits. However, each packet is selfdecodable. This characteristic is achieved by sending the systematic bits (info initial bits) in every retransmission required.

This solves the following scenario, already presented in above sections. If systematic bits are only included into the initial attempt, and this packet finds very bad channel conditions and it is received so damaged, there will not be a chance to correctly decode the information even though a lot of new redundant bits are sent.

As it was explained, this problem can be solved by adding some signalling, allowing the receiver to decide either to ask for a retransmission including new redundant bits or to ask for a retransmission where systematic bits are included again. However, the type 3 H-ARQ protocol explained here is considered without feedback, and just sending ACKs or NACKs is allowed, so the set of bits to be sent are decided independently from the receiver, and the systematic bits are included every time.

Figure 13 exemplifies this process.

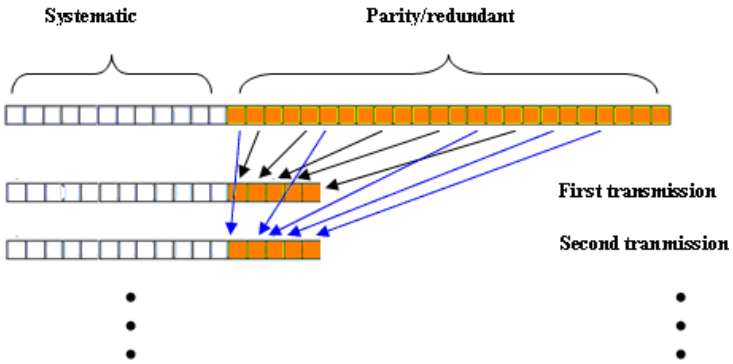

Figure 13 *Example of a type 3 H-ARQ process*

The importance of sending the systematic bits every time is clear. However, the system throughput decreases by doing it. Therefore, it becomes a trade off between the safety measure of sending systematic bits every time and the throughput achieved.

3.3. Different H-ARQ Schemes Comparison

Comparisons between the different H-ARQ schemes performance can be found in the 5.2.3. Results from simulations using OpenWNS will be compared with the results presented in the literature.

3.4. History

H-ARQ protocols started being used in the HSPA release, and they will be used into next generation of mobile communications.

The H-ARQ concepts and types presented up to now reflected the main ideas and characteristics, and they are completely valid to analyze the present and the future of H-ARQ protocols. Therefore, in order not to repeat concepts already presented, only the details belonging to the implementation in each technology, past and future, will be present from now on.

3.4.1 HSPA

H-ARQ is used in HSPA to provide robustness against occasional transmission errors and improve the link efficiency to increase the capacity.

3.4.1.1 HSDPA

The following lines will present some interesting details about the H-ARQ implementation in the HSPA downlink [12].

The H-ARQ functionality covers both the Media Access Control (MAC) and the PHY. Since the MAC is located in the Node B (base station), erroneous transport blocks can be rapidly retransmitted, representing a gain in terms of delay due to retransmissions compared with Radio Link Control (RLC) retransmissions.

The node B decides whether to use Incremental Redundancy or Chase Combining scheme by choosing the puncturing pattern to be used for the retransmission (the exactly same set of bits as the initial transmission for CC or a different set of bits for IR).

It is been proved that the IR scheme provides significant gains when the initial code rate is high [14]. More details can be found in 5.2.3. Thus, node B usually decides to use IR rather than CC when the User Equipment (UE) is close to it, and the transmission power does not limit the achievable data rate.

In HSDPA the UE is the one asking for retransmissions. Once a packet has been received, the UE tries to decode it. Depending on the success or not of the decoding, he confirms the Base Station (BS) a good reception or asks for a retransmission (both actions by sending a bit).

In HSDPA retransmissions are scheduled as any other data and the Node B is free to schedule the retransmission to the UE at any time instant and using a redundancy version of its choice. This type of operation is often referred to as adaptive asynchronous hybrid ARQ. Adaptive since the Node B may change the transmission format and asynchronous since retransmissions may occur at any time after receiving the ACK/NAK. Figure 14 exemplifies the process.

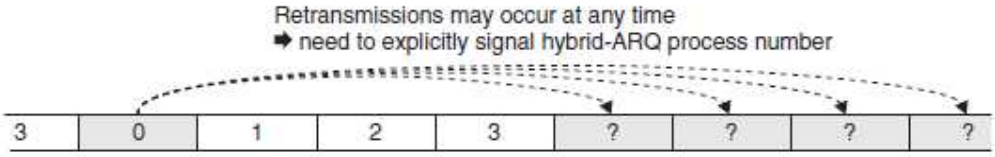

Figure 14 *Asynchronous H-ARQ [12]*

In order for the UE to be able to decode the information correctly, he needs to know whether the packet received is a retransmission of previous data transmitted or a transmission of new data. To solve this challenge, some signalling is included into the downlink transmissions. As it can be read in the figure above, the downlink packet must specify the H-ARQ process number, allowing the UE to decide whether this data must be combined with previous data or it has to be decoded on its own.

The H-ARQ structure implemented in HSDPA is a stop and wait scheme. In order to support continuous transmission to a single UE, multiple stop and wait structures are used in parallel. Each of these H-ARQ processes has its own buffer, identified by a number, where erroneous transmissions are stored while the new retransmission arrives. The data is deleted from the concrete buffer once the decoding of this block has been successfully performed.

One result of having multiple independent H-ARQ processes operated in parallel is that decoded transport blocks may appear out-of-sequence. As the RLC protocol assumes data to appear in the correct order, a reordering mechanism is used between the outputs from the multiple H-ARQ processes and the RLC.

3.4.1.2 HSUPA

The implementation of H-ARQ in the uplink is similar to the one presented in the downlink.

Now, the Node B is the one informing the UE about the successfully reception or not, asking for a retransmission if necessary.

One of the main differences between the uplink and the downlink H-ARQ implementation concerns the use of soft handover in the uplink. When a UE is in soft handover, the H-ARQ protocol needs to be performed by various BS. The data transmitted by the UE can be received in one BS but not in another one. Regarding the

UE, it is sufficient if at least one BS successfully receives the information and sends an ACK. Therefore, a retransmission is only sent when no ACK is received from at least one of the BS.

For the uplink, differently from the downlink, retransmission follows a non-adaptative synchronous pattern, where retransmissions take place at a predefined time after the initial transmission, and the different possible puncturing sets of bits are predefined from the moment of the initial transmission. That leads to an afford of control signalling overhead, since there is no need to specify the H-ARQ process number. The next figure exemplifies this.

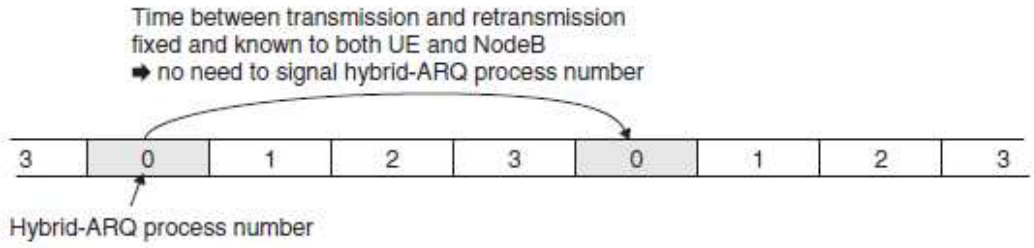

Figure 15 *Synchronous H-ARQ [12]*

The reordering process can not be located in the node B due to the soft handover. Several node B can be involved in an H-ARQ process. Therefore, the reordering must be performed in the RNC (radio network controller).

3.4.2 LTE

H-ARQ to be present on LTE technology has been already designed. It is similar to the H-ARQ mechanism for HSPA. The purpose continues being to provide robustness against transmission errors, at the same time it is a tool for increase capacity.

Like it was explained in HSPA, H-ARQ spans both MAC and physical layer. The combining bits process is performed by the physical layer.

The structure designed for LTE is similar to the one for HSPA, consistent in a multiple parallel stop and wait processes.

Again, the receiver must be aware of whether the block received is a retransmission or a new transmission. Moreover, it must know to which H-ARQ process the block belongs. This is solved with the use of signalling.

Similarly to HSPA, an asynchronous protocol is used for the downlink H-ARQ operations, while a synchronous protocol is used for the uplink. Hence, downlink retransmissions occur at any time after the initial transmission, and a number is used to indicate to which H-ARQ process is addressed. Uplink transmissions, differently, occur a predefined time after the initial transmission, and the process number can be implicitly derived.

Due to the characteristics explained, H-ARQ is not applicable for all types of traffic. Broadcast transmissions, where the information is addressed to multiple users, do not use H-ARQ. This last statement is valid for all kinds of H-ARQ studied up to know.

CHAPTER 4

Link-to-System Interface in OpenWNS

After the theoretical study of the L2S interface and its mapping mechanisms, it is time to see how these techniques have been implemented in OpenWNS and which performance they provide.

4.1. Introduction

4.1.1 Effective SINR mapping

MIESM was the L2S ESM technique chosen to be implemented in OpenWNS. More specifically, RBIR approach. The reason: all the literature dedicated to compare the different approaches concluded that MIESM provides the most accuracy results in almost every possible scenario.

In order to support this state, the following tables show the results obtained with the simulations run in [1], comparing the performance of EESM and MIESM using diverse MCS. The parameter used to evaluate the comparison is σ, the root-mean-square of the difference between the PER measured and the PER obtained with the simulation.

$$\sigma = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(PER_{\mathrm{model},n} - PER_{measured,n})^2} \qquad (4.1)$$

N is the number of values taking into account.

Table 2 Different MCS used within the simulation in order to compare the performance of diverse ESM.

MCS	Coding Scheme	Modulation	Code Word Length
1	Convolutional Code 1/2	QPSK	408
2	Convolutional Code 1/2	16QAM	824
3	Convolutional Code 3/4	16QAM	1240
4	Turbo Code 1/3	16QAM	544

Table 3 σ values obtained using different ESM approaches and MCS

MCS	ESM Approach	σ
1	EESM	0.020
	MIESM	0.016
2	EESM	0.040
	MIESM	0.013
3	EESM	0.027
	MIESM	0.013
4	EESM	0.009
	MIESM	0.010

It can be seen MIESM outperforms EESM in the first three scenarios, whereas EESM works better in the last one, where the code rate is lower than in the first three scenarios. It is been shown that EESM works better with lower code rates, whereas MIESM obtains the better results the higher the code rate used [1], [4]. Even though the values achieved with the last MCS, the difference between approaches is so small that it can be concluded that MIESM offers better performance in general.

4.1.1.1 MIESM theoretical computation

When the MIESM was theoretically presented in 2.2.1.3, the mathematical concept of MI was introduced. The MI value calculated in MIESM, however, does not exactly fit the idea of how dependent two variables are, but how similar they are. For example, it somehow compares the packet transmitted and the packet received, and provides a value that represents how similar both packets are.

Another way to understand the same concept would be the following. The MI values represent how much information arrives to the receiver.

In order to clarify those ideas, formula (4.2) represents how the symbol information, SI, is calculated. This is actually not the formula used to calculate the MIB, but it helps to understand the concept, and the relation between MIB and SINR. Further details about how MIB is really calculated, as well as numerical examples, can be found in Appendix A.

$$SI(SINR_n, m(n)) = \log_2 M - \frac{1}{M}\sum_{m=1}^{M} E_u\left\{\log_2\left(1+\sum_{k=1,k\neq m}^{M}\exp\left[-\frac{\left|X_k - X_m + U\right|^2 - \left|U\right|^2}{\frac{1}{SINR_n}}\right]\right)\right\} \quad (4.2)$$

Where U is a zero mean complex Gaussian with variance $1/(2\,SINR_n)$ per component, $SINR_n$ is the post-equalizer SINR at the n_{th} symbol or sub-carrier, M is the order of

the modulation, $m(n)$ is the number of bits at the n_{th} symbol (or sub-carrier) and X_i is the i bit into the symbol.

The main idea is the following. The maximum information that a symbol can provide once it has been received is the number of bits it has. However, and due to channel conditions, the information could have been damaged. The value used to represent those channel conditions and the damage caused is the received coded block SINR, as it was discussed in 2.1.3. Bad channel conditions are represent by low SINR values. As it can be seen from (4.2), the lower the SINR, the higher the value subtracting the initial $\log_2 M$, thus lower SI. Contrary, the higher the received SINR value, representing good channel conditions, the lower the value subtracting $\log_2 M$, thus higher SI.

Assuming now N sub-carriers are used to transmit the coded block, the mutual information per received bit (MIB) is given by:

$$MIB = \frac{\sum_{n=1}^{N} SI\left(SINR_n, m(n)\right)}{\sum_{n=1}^{N} m(n)} \tag{4.3}$$

Further details of the mathematical derivations can be found in [6] and Appendix A.

4.2. Link-to-System Mappings Implementation

Let's summarize for the last time the objective of the L2S mapping mechanisms before presenting how OpenWNS performs it. As it was said, Mutual Information Effective SINR Mapping is used for the Effective SINR Mapping.

The final goal of the OpenWNS implementation regarding the L2S interface is to provide a BLER given a set of configuration parameters and instantaneous predicted random conditions, like traffic, interference, fading, scheduling, etc. Those parameters are used to predict the received coded block SINR. This SINR value is used then to calculate a MI value.

The theory says [6] that once these MI values are obtained, an effective SINR can be calculated, coming back to the SINR domain. However, as it was mentioned, OpenWNS does not calculate this effective SINR value, and it just keeps the MI values in order to realize the MI to BLER mapping and provide the desired BLER value.

Figure 16 represents the computational procedure of MIESM and MI to BLER mappings, and it will help to understand how both mappings have been implemented in OpenWNS.

The ESM is located on the left part of the picture. It goes from SINR values to MI values, which can be easily and fairly averaged in order to obtain an effective MI value. It can be seen that the result depends on the modulation scheme used. The actual dependence is explained later.

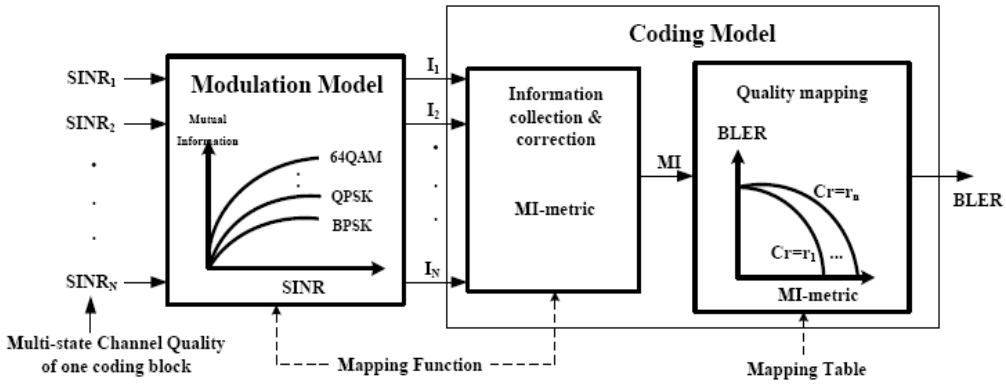

Figure 16 *Computational procedure for MIESM method [6]*

The MI to BLER mapping, located on the right side of the picture, goes from the MI values to a BLER value. Again, the modulation coding scheme influences the result, as well as the block length.

The computation of MI values from SINR values is not trivial, and in the hypothetical case it wanted to be calculated mathematically every time, it would represent a computationally cost too high. The same would happen to calculate the BLER given a MI value.

The solution to these problems is solved with the use of stored look-up tables. Those tables relate SINR to MI values and MI to BLER values depending on the parameters that have influence in each mapping. Therefore, both ESM and the MI to BLER mapping are finally performed in OpenWNS as a simple searching process into a table. Further explanations about the tables can be found in the following sections.

4.2.1 Tables Available

The origin of the tables and how they were calculated can be found in 4.3.1.1 and 4.3.2.1.

OpenWNS wants to provide the researcher the possibility to adapt the simulations according to his needs. Regarding MIESM and MI to BLER mapping, that means being able to used different modulations coding schemes and block lengths, which means being able to use stored tables with those parameters. However, not every possible combination can be chosen yet. The tables already filled and available in OpenWNS are:

i) For the first map, tables using BPSK, QPSK, 8QAM, 16QAM, 32QAM and 64 QAM.

ii) For the second map, the coding available is Turbo Code UMTS, using 1/3, 1/2, 2/3 and 5/6 code rates. Different BL are available for each CR. If a different BL from one of the stored tables is used within a simulation, OpenWNS finds the suitable one.

4.3. Link-to-System Mappings Evaluation

The L2S interface was already implemented in OpenWNS before this thesis started. Then, this thesis will just evaluate the code implemented, validating its performance.

The evaluation is organized as follows: both mappings, from SINR to MI and from MI to BLER, will be further studied from now on, focusing on specifying how the mapping tables were calculated and understanding how the inputs of the mappings determine the outputs.

4.3.1 Effective SINR Mapping

The Effective SINR Mapping chosen to be implemented in OpenWNS is Mutual Information ESM.

As it was explained, MI values are not calculated mathematically every time, due to the cost that this calculation would take. Instead, stored tables relating SINR and MIB values are used.

As it has been reference along the thesis, the mathematical procedure to calculate the MI given the SINR and the modulation scheme is shown in Appendix A.

4.3.1.1 Origin of the Tables

[6] encloses some tables relating SINR and MIB values depending on the modulation used, calculated using RBIR method, studied in Appendix A. Table 4 presents some of the results published in [6] (left table), and the corresponding values obtained by looking at OpenWNS tables (right table).

Table 4 SINR to MIB values for different modulations included in [6] (left). Same values from OpenWNS tables (right)

[6]	QPSK	16QAM	64QAM
SINR (dB)	**MIB**	**MIB**	**MIB**
-20	0.0072	0.0036	0.0024
-10	0.0688	0.0344	0.229
0	0.4859	0.2474	0.1653
10	0.9968	0.7910	0.5448
20	1	1	0.9668

OpenWNS	QPSK	16QAM	64QAM
SINR (dB)	**MIB**	**MIB**	**MIB**
-20	0.00717	0.00358	0.00239
-10	0.06874	0.03437	0.22916
0	0.48594	0.24743	0.16529
10	0.99675	0.79097	0.54476
20	1	0.99997	0.96691

OpenWNS tables provide MI values as a function of SINR and the modulation scheme used. Therefore, the value has been divided by the order of the modulation in each case to obtain a MIB value.

As it can be seen, values in both tables are the same. Hence, it can be assure OpenWNS tables for ESM have been calculated following the RBIR method in [6], and analyzed in Appendix A.

The next figure depicts the SINR to MI. Values from OpenWNS have been used, but as it has been explained, the same graphic would have been obtained in case values from [6] had been depicted.

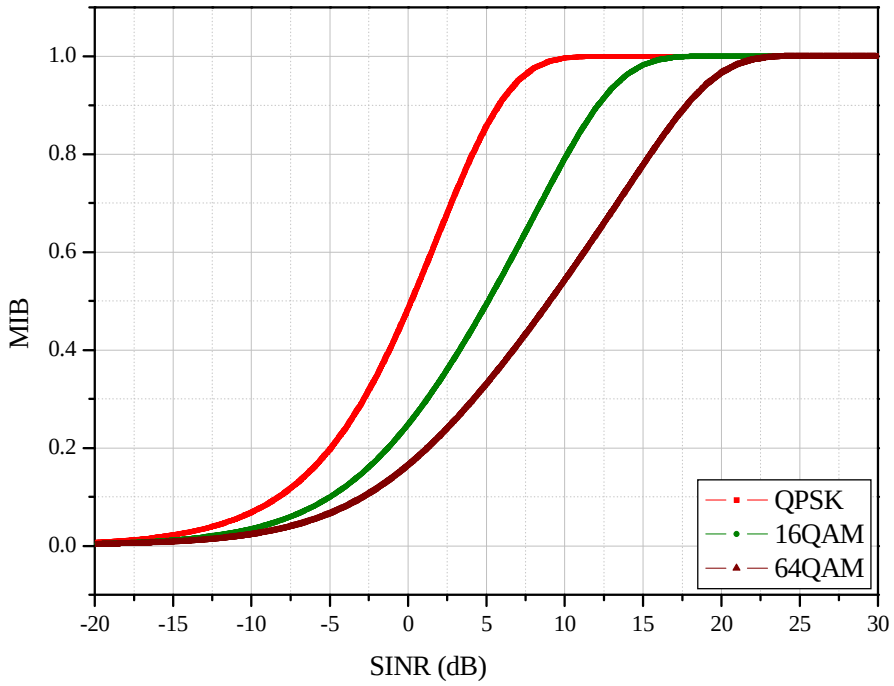

Figure 17 *SINR to MIB relation for different modulation using tables in OpenWNS*

Once the precedence of the tables used by OpenWNS regarding the ESM is clear, it is time to study how the inputs determine the output of the MIESM.

4.3.1.2 Influence of the Parameters in MIESM

The MIB value resulting from the MIESM depends on the SINR and the modulation scheme used.

$$MI = f\left(SINR, Modulation\right) \tag{4.4}$$

4.3.1.2.1 SINR

System level simulations provide a predicted SINR. This value represents the link layer behaviour, and consequently, it will determine how good the information has been received.

Remembering now the MI concept given in 4.1.1.1, a MI value represents the similarity between the data sent and the data received. Therefore, high MI values will be obtained when the blocks are received with high SINR, since the link layer had not corrupted so many bits, and the data received will be similar to the data sent. Contrary, a low received SINR means the channel might have hardly damaged the data, and the received block will be more distant to the block sent.

The relationship between SINR and MIB explained above has been already depicted in Figure 17 , and it can be understood too by looking at (4.2).

4.3.1.2.2 Modulation

The reason why the modulation scheme determines the MI value was already presented in 2.2.1.1. Basically, the modulation scheme determines the distance between symbols within a constellation. The higher the order of a modulation, the closer the symbols are within the constellation. Due to this, it is more difficult to distinguish between symbols, and less Information can be recovered from each symbol.

Next figure illustrates the relation between the received SINR and the MIB, comparing different modulations.

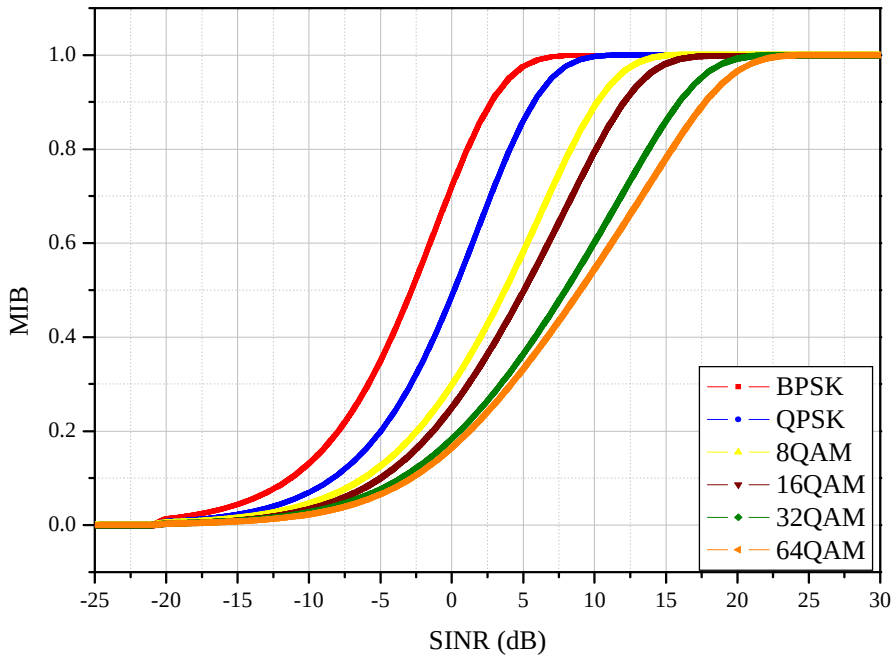

Figure 18 *SINR to MIB relation for different modulation schemes*

It can be observed that the higher the order of the modulation (more bits per symbol), the higher the SINR to achieve a specific MIB value, as it was predicted. As an example, given a SINR of 5dB, a MIB close to 1 is obtained if the modulation used is BPSK, meaning the block received is almost equal to the block sent. Instead, less than 0.4 information per bit is obtained if a 64QAM is used. That would mean that almost two thirds of the block has been damaged and corrupted.

4.3.2 Mutual Information to BLER Mapping

This mapping goes from a MIB value into a BLER value. The modulation coding scheme and the block length influence the result.

However, 2.2.2.1 further analyzed this mapping, and it was said that the influence of the modulation alphabet is almost irrelevant. Thus, the afford in terms of complexity and computationally cost gained by discarding the modulation scheme as a parameter into the second mapping is preferable. Therefore, only the MIB value, the CR and the BL used influence the MIB to BLER mapping.

4.3.2.1 Origin of the Tables

As it has been mentioned along the thesis, the results relating MIB values and BLER values are normally obtained by running sophisticated link level simulations. However, in 2.2.2.1 a mathematical approach was presented.

Figure 19 depicts the MIB to BLER relation for different MCS and BL. It also shows the difference between the results obtained by using OpenWNS tables, and the results obtained by using the parametric function that fits the AWGN BLER curves, presented in 2.2.2.1, and proposed in [6].

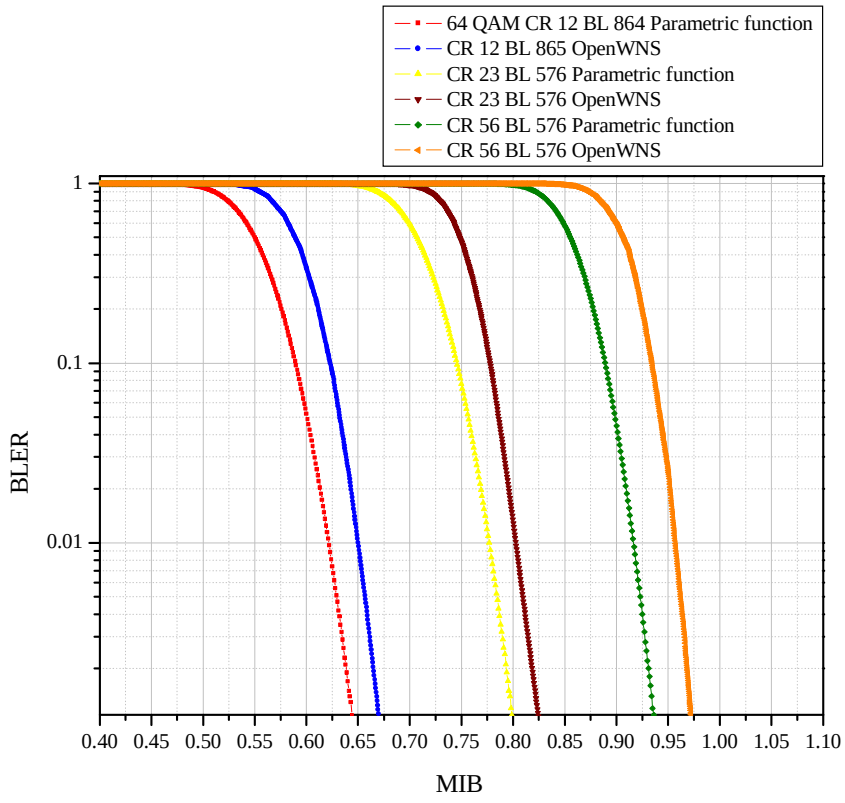

Figure 19 *MIB to BLER for different MCS and BL, comparing OpenWNS and parametric function performance*

As it can be seen, the results differ in 0.05 bits MI approximately. The reason seems to be the coding scheme used within the different approaches. All the simulations run

in [6] and the results obtained are determined by the coding scheme assumed along this text, Convolutional Turbo Coding (CTC). Differently, OpenWNS uses Block Turbo Codes (BTC).

It has been proved that CTC outperforms BTC for BLER lower than 10^{-5} [15], presenting a gain of around 0.5 to 0.7 dB. This is more or less the SINR difference that corresponds to a 0.05 MIB difference, as it can be seen by looking at OpenWNS tables. Further details about CTC, BTC are given in Appendix D.

4.3.2.2 Influence of the parameters in MI to BLER mapping

Like it has been presented for ESM, the study of how the inputs affect the output value can be found from now on regarding the MI to BLER mapping.

$$BLER = f\left(MIB, BL, MCS\right) \tag{4.5}$$

4.3.2.2.1 Mutual Information per Bit

MIB values come from the first mapping, where it is been presented that the higher the SINR, the higher the MIB. Therefore, the MIB value, as SINR does, gives us an idea of how good the transmission has been, how many information can be recovered, and thus it will determine the received BLER.

A high MIB means higher similarity between the data transmitted and the received. In other words, there are no so many errors, and the chances to successfully decoding the data are good. That scenario is represented by a low BLER value. Contrary, a low MIB value means there are more corrupted bits within the received block with resptec of the block sent. Consequently, it might happen that even with the use of FEC, the data can not be successfully recovered. This scenario is represented by a high BLER value.

Figure 20 depicts this relation, where the CR and the BL used for each modulation is the same, and thus irrelevant to understand the relation between MIB and BLER. Three modulations are compared.

It can be appreciated that the behaviour is the same for each modulation. This is coherent with the assumption taken in OpenWNS implementation, which says that the modulation scheme is not an input into the MI to BLER mapping, and consequently there should not make any difference. As it was said in 2.2.2.1, that assumption is taken after studying that the differences that the modulation scheme causes is enough small to be discarded.

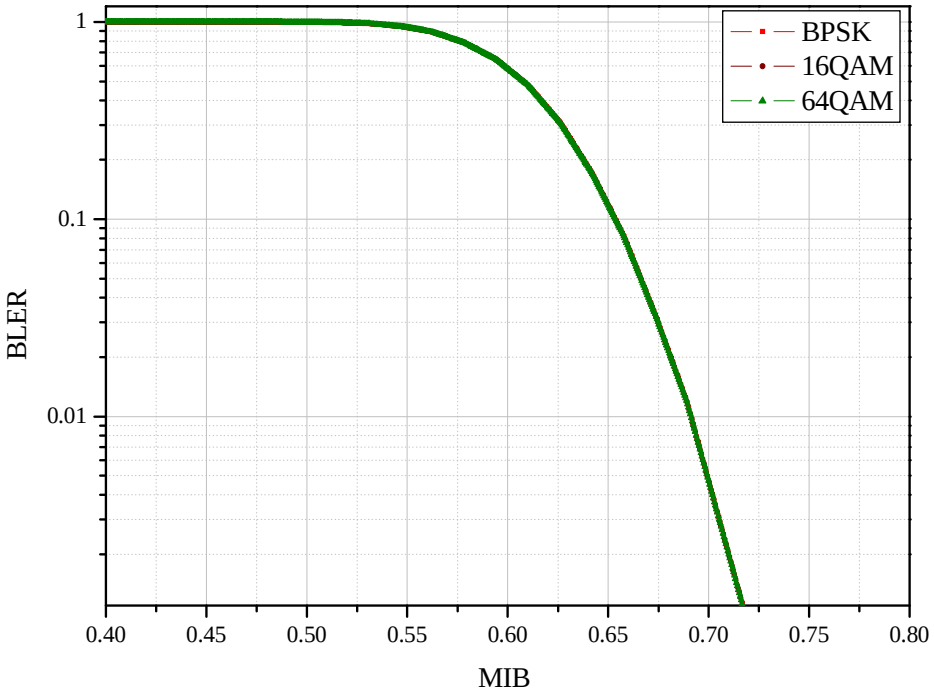

Figure 20 *MIB to BLER relation for different modulation. CR=1/2, BL=256 bits.*

However, it is still interesting to study how the CR and the BL affect the MI to BLER mapping depending on the modulation used. Therefore, from now on graphics regarding MIB to BLER mapping will be expressed as a function of SINR, and not MIB. By doing this, differences between modulations alphabets when the CR and the BL change will be appreciated.

Moreover, it seems easier to understand an increment or a decrease in terms of a SINR required to achieve a BLER than using a MIB difference.

4.3.2.2.2 Coding Scheme

i) Code Rate

The CR states what portion of the total amount of information represents the payload and which represents the redundant bits added to fortify the code. Lower code rates mean more protection, more redundant bits added for each payload bit. Consequently, the chances of successfully decoding the information increases, and the received BLER decreases.

The next figures depict the SINR to BLER obtained using different code rates. Three different modulations schemes are depicted, in order to evaluate how the increasing or decreasing the code rate affects each one. The block length used is the same for the three figures, and thus irrelevant to the interesting comparison in here.

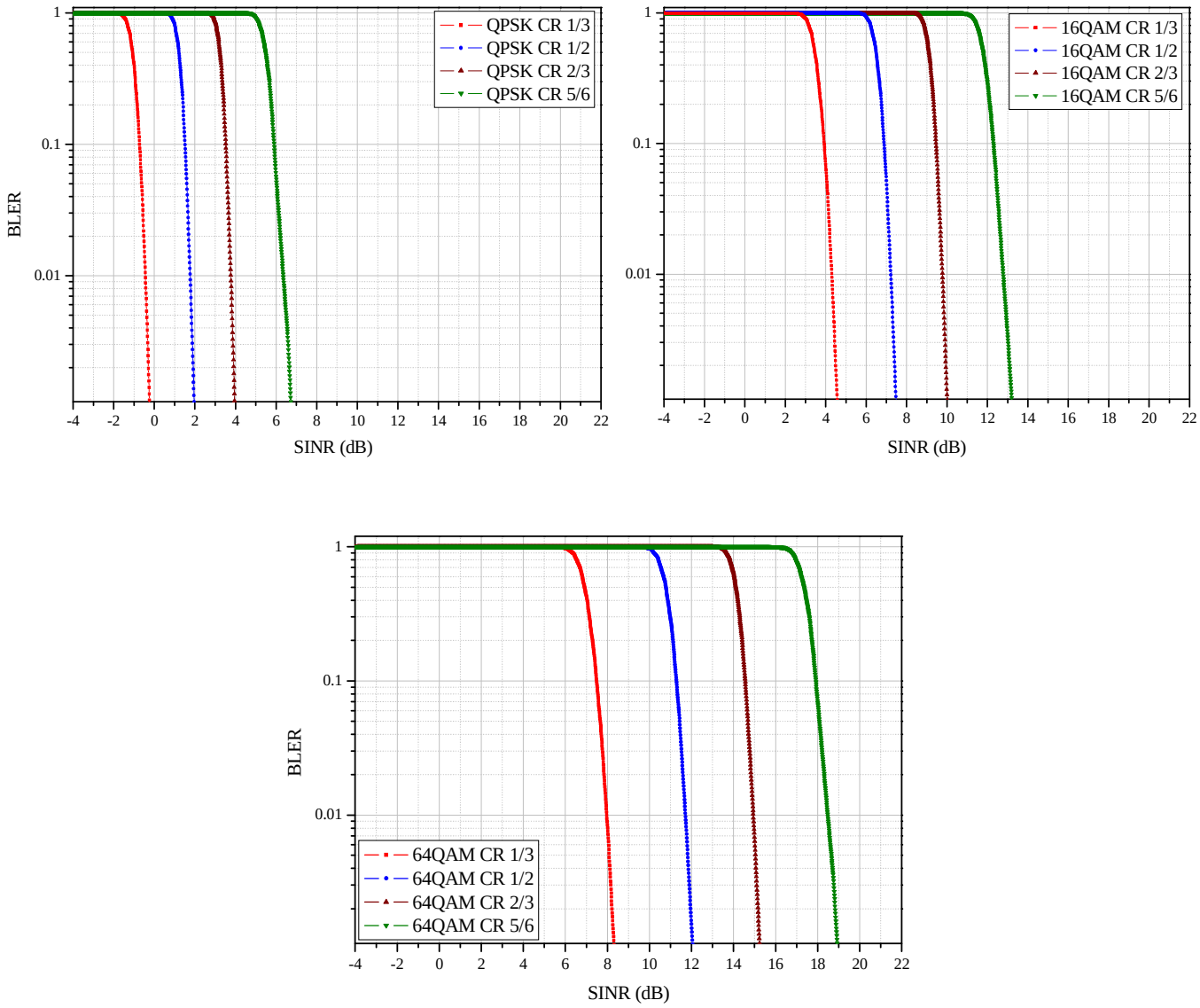

Figure 21 *SINR to BLER relation for QPSK, 16QAM and 64QAM using different CR. BL=1024 bits.*

It can be seen that the lower the code rate (more redundant bits for each payload bit) used, the lower the SINR required to achieve a target BLER. Contrary, higher SINRs are required to achieve the target BLER when higher code rates are used.

By looking at these results, it seems that using lower code rates provides a better performance. This is true in terms of BLER, but it is completely the contrary when improving the throughput is the goal to achieve. When using a lower code rate, more redundant bits are sent instead of payload bits. Consequently, the effective throughput decreases. In conclusion, choosing one code rate or another is a trade off between the security of the transmission, represented by the BLER, and the final effective throughput.

The SINR offset comparing the three modulations comes from the first mapping, due to the relation between MI and SINR. Higher order modulations needed higher SINR to achieve the same performance, in that case the same BLER. Thus, it is not related with the code rate used.

The behaviour depicted in Figure 21 seems to be, in general, the same for the three modulations. However, it can be interesting to see whether the differences in terms of SINR remain constant or not when changing the modulation alphabet. The next table studies this behaviour.

Table 5 SINR values to achieve a BLER = 0.1 for different modulations and different CR. BL=1024 bits. *Inc* denotes the increment in terms of SINR when using different CR.

Desired BLER ≈ 0.1	SINR CR=1/3 (dB)	*Inc*	SINR CR=1/2 (dB)	*Inc*	SINR CR=2/3 **(**dB**)**	*Inc*	SINR CR=5/6 **(**dB**)**
QPSK	-0.75	**2.25**	1.5	**2**	3.5	**2.3**	5.8
16QAM	3.8	**3**	6.8	**2.6**	9.4	**2.8**	12.2
64QAM	7.5	**3.75**	11.25	**3.25**	14.5	**3.4**	17.9

It can be appreciated that increasing the CR by 1/6 causes an increment between 2 and 3.5 dB in terms of SINR required to achieve a BLER about 10%.

It can be seen too that for higher modulations, the increment in terms of SINR required when the CR increases is bigger than for lower modulations.

ii) Coding

OpenWNS uses turbo UMTS coding. Not comparison can be done up to know, since no more coding schemes are available. Some information about Turbo Codes is given in Appendix D.

iii) Block Length

The first idea would be that the longer the BL, the more bits can suffer damage, thus bigger chances to fail in general.

However, turbo codes, like the used in UMTS, works better the longer block length used [2]. Therefore, the relation between BL and BLER has to be considered together with the coding used. Since OpenWNS uses turbo UMTS code, the longer the BL, the lower the BLER achieved given the same SINR.

Figure 22 depicts this fact, comparing the BLER as a function of SINR given different modulation schemes and using different BL. Note that the CR used for the three graphics is the same, so it does not have influence on the comparison results.

As it happened when studying how the CR influences the result into the MI to BLER mapping, the offset in terms of SINR when comparing different modulations comes from the first mapping and it must not be taken into account right now.

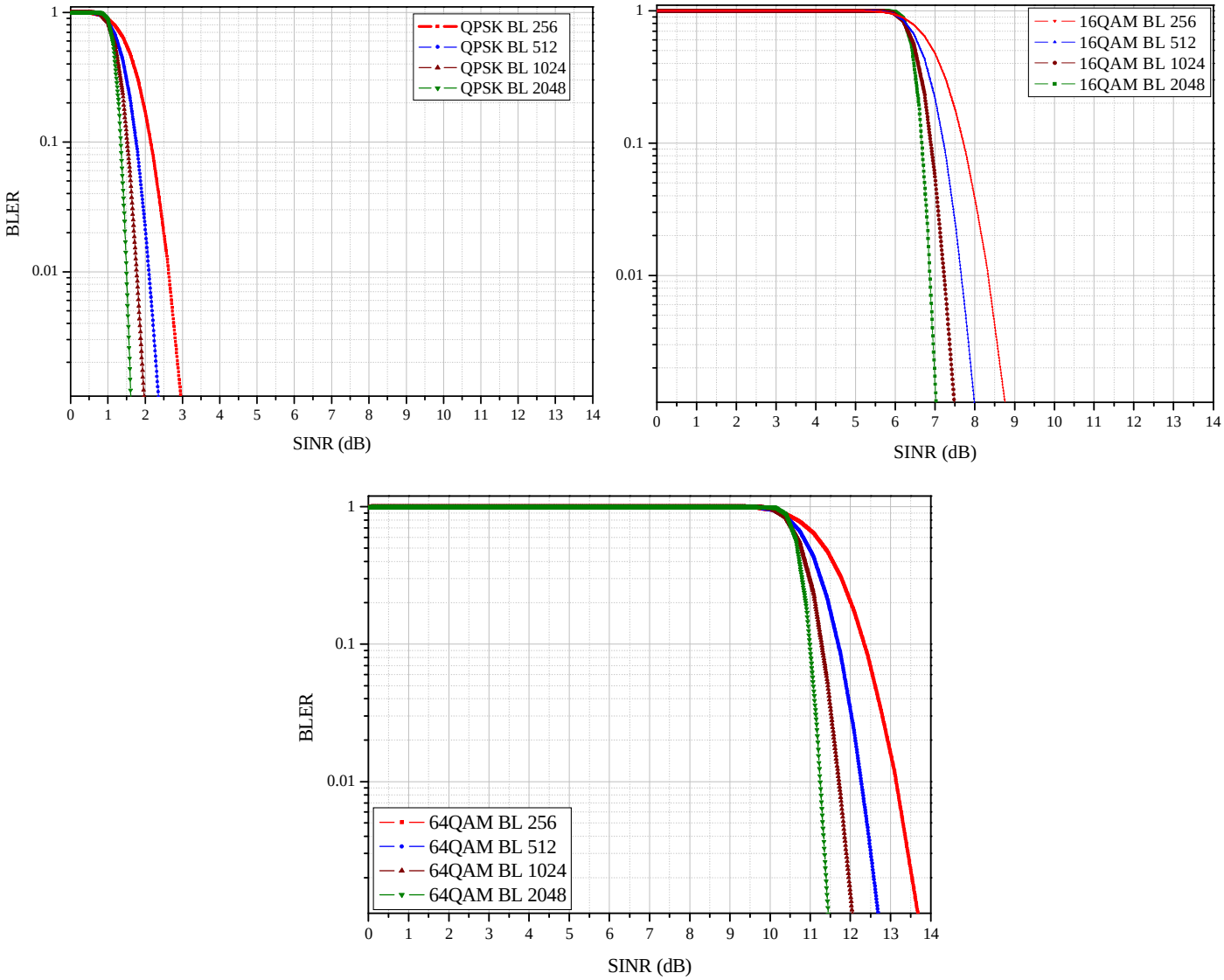

Figure 22 *SINR to BLER relation for QPSK, 16QAM and 64QAM using different BL(bits), CR=1/2.*

The behaviour is the same for the three modulations. The higher the BL, the lower the SINR required to achieve the desired BLER, as it was expected. As it was done when studying the influence of the CR, it is interesting to see how the differences in terms of SINR are depending on the modulation used. Next table shows that difference.

Table 6 SINR values to achieve a BLER = 0.1 for different modulations and different BL. CR=1/2. $|Inc|$ denotes the absolute value of the increment in terms of SINR when using different BL.

Desired BLER ≈ 0.1	SINR BL=256 (dB)	$\|Inc\|$	SINR BL=512 (dB)	$\|Inc\|$	SINR BL=1024 **(**dB**)**	$\|Inc\|$	SINR BL=2048 **(**dB**)**
QPSK	2.15	**0.39**	1.76	**0.24**	1.52	**0.18**	1.34
16QAM	7.7	**0.5**	7.2	**0.3**	6.9	**0.25**	6.65
64QAM	12.35	**0.45**	11.7	**0.65**	11.25	**0.25**	11

It can be seen that the higher the modulation used, the bigger the SINR differences when the BL changes its value. Again, like it happened with CR, higher modulations are more sensitive to changes.

4.3.3 ESM and MI to BLER mapping together

Once both mappings have been studied and tested it is possible to evaluate the performance of both mappings working together one after the other.

Figure 23 shows the expected throughput achieved depending on the modulation coding scheme used. The BLER value required to perform the calculation has been obtained by using the L2S mapping mechanism implemented in OpenWNS and tested up to now. The same code rate and block length have been used for each modulation to run this test, affecting the final throughput value, but not affecting the modulation comparison.

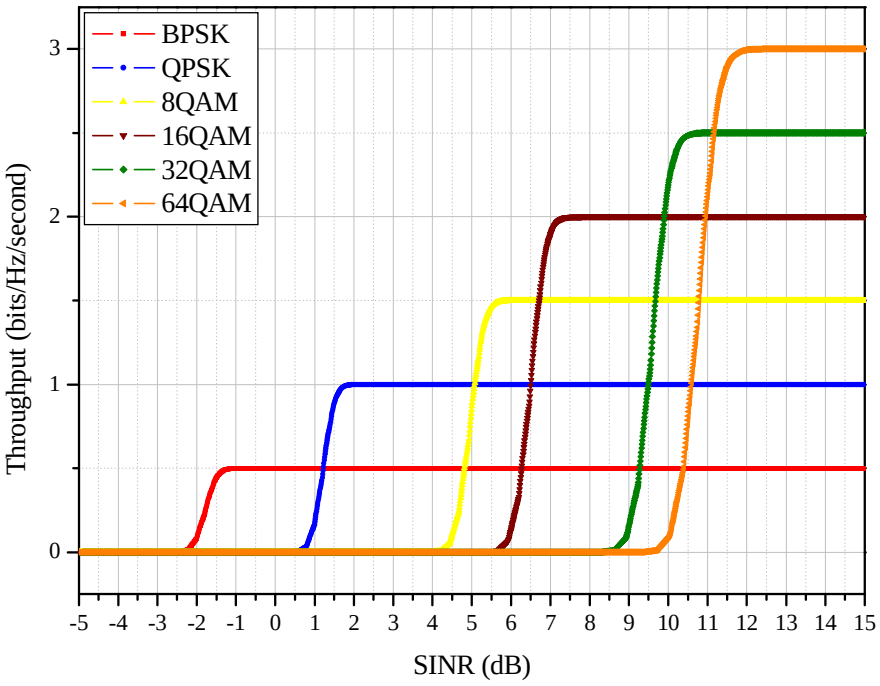

Figure 23 *SINR to Throughput relation for different modulations. CR=1/2 and BL= 1024.*

It is interesting to see how these curves will show up when H-ARQ protocols are introduced in OpenWNS. Predicted results can be seen in 5.2.1.3.2 and 5.2.2.3.2.

4.4. OpenWNS versus WiMAC L2S Interface

The reasons why the L2S interface and its mapping mechanisms studied along this thesis were needed were presented back in 2.2. It is been further analyzed too how these mapping functions allows the simulator to obtain a BLER value from a set of channel quality measures. However, the SINR to BLER mapping has not always been calculated like it has been shown along this thesis. Back in the past, where no instantaneous channel conditions were used to predict the link layer performance, a direct map from a SINR to a BLER was used.

In order to compare the performance provided by the L2S interface implemented in OpenWNS and the performance obtained with the techniques used before, it is time to present WiMAC.

WiMAC, like OpenWNS, is a software-based simulator with a prototype implementation of the IEEE 802.16 protocol developed at ComNets institute, RWTH Aachen University.

The differences between both simulators are significant. However, this thesis will just evaluate the accuracy provided by the new L2S interface implemented in OpenWNS.

First of all, let's study how WiMAC calculated a BLER value.

Similarly to OpenWNS simulator, a channel model provided a received SINR value for a particular packet. The SINR value was directly mapped into a BLER value, using a look-up tables like the ones used in OpenWNS, but without the intermediate MI step. Those tables were generated by a sophisticated link layer simulation chain develop during the IST-STRIKE project [17].

Since the packet length simulated in WiMAC was not fixed, the resulting BLER was calculated based on smaller units like bits, bytes or OFDM symbols. Simulations showed that errors of OFDM symbols were not correlated [17], thanks to the interleaving performed. Thus the OFDM symbol error ratio was used as the interface to the protocol simulator.

In the protocol simulation, the resulting packet error ratio of a PDU (BLER in the formula) was calculated as follows. The calculation was based on the PDU size NOFDM measured in OFDM symbols and the OFDM symbol error ratio (pOFDM).

$$BLER = 1 - (1 - pOFDM)^{NOFDM} \tag{4.6}$$

Figure 24 shows the BLER achieved as a function of the SINR for different MCS, using the values of the stored tables used in WiMAC. Note that in those tables the value available was $pOFDM$. Therefore, (4.6) has been used in order to achieve a BLER to be compared with the one obtained with OpenWNS.

The BL used for the first simulations is 256 bits. Therefore, NOFDM is 256/2 for QPSK case, 256/4 for 16QAM and 256/6 for 64QAM.

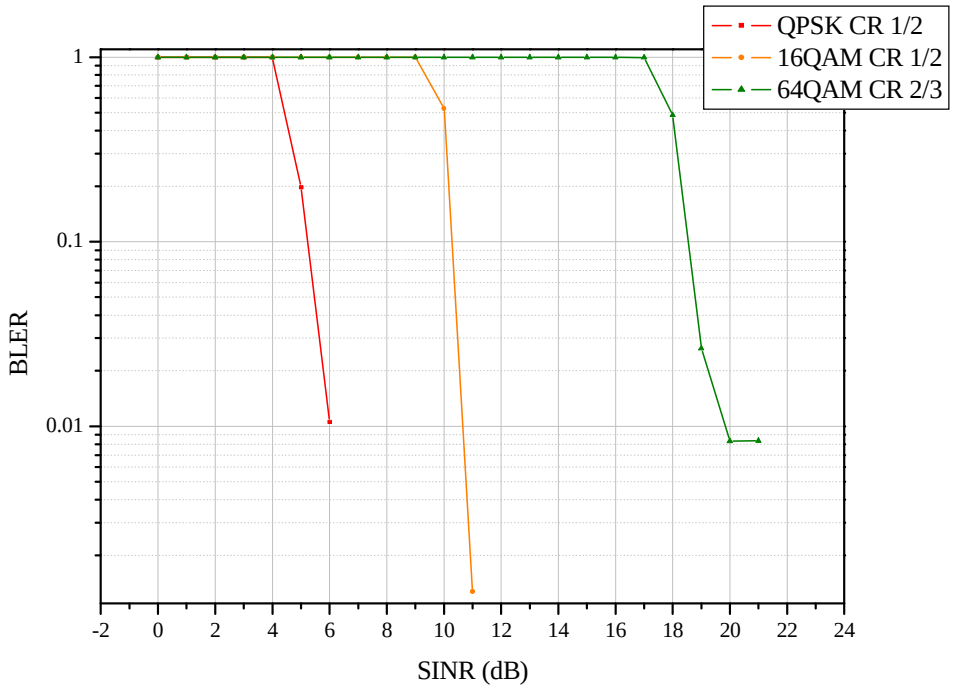

Figure 24 *SINR to BLER relation for different MCS. BL=256 bits. Values from WiMAC tables*

Next figure shows the same relation depicted above, but using the L2S interface implemented in OpenWNS. Again, the BL used is 256 bits.

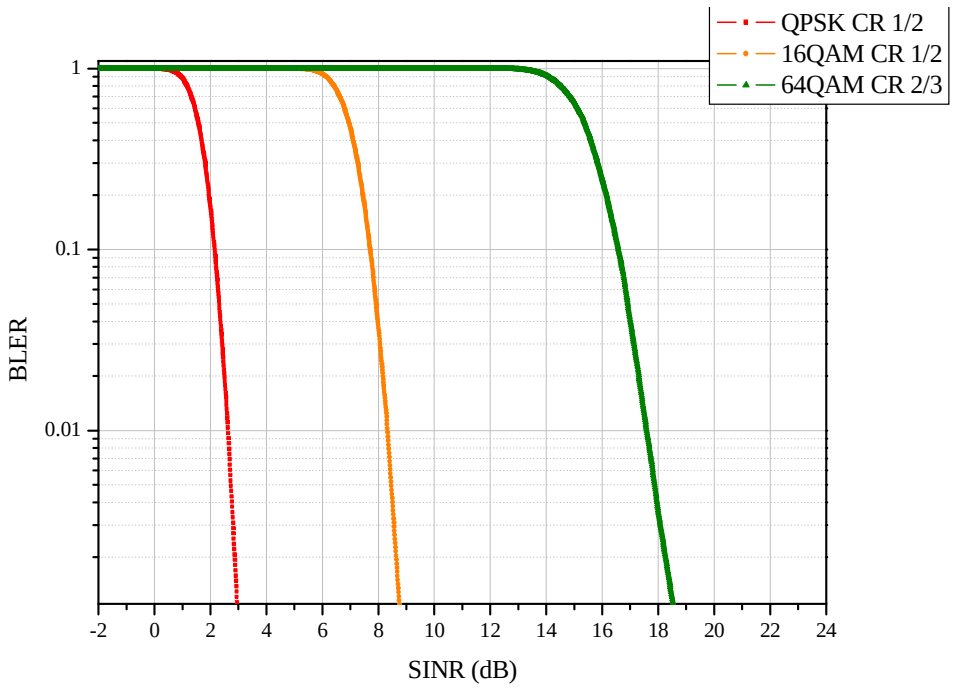

Figure 25 *SINR to BLER relation for different MCS. BL=256 bits. Values from OpenWNS tables.*

Next table shows some of the values used to run the graphics, and it will help to understand the differences between the two implementations.

Table 7 SINR values when BLER=0.1 for different MCS and BL=256, using values from WiMAC and OpenWNS. Comparative.

	SINR WiMAC	SINR OpenWNS	Increment/Decrement
QPSK CR 1/2	5.2 dB	2 dB	3.2 dB
16QAM CR 1/2	10.2 dB	7.6 dB	2.6 dB
64QAM CR 2/3	18.5 dB	16.5 dB	2 dB

It can be seen then that OpenWNS presents an estimation of around 3 dB better on average respects from WiMAC, and the way the BLER was predicted or calculated before the new mechanisms were designed. It should say then that the performance in WiMAC was underestimated.

It can be interesting to study the evolution of this difference when using another BL.

Figure 26 shows the BLER value as a function of SINR and MCS achieved using WiMAC tables. The BL used this time is 1024. Again, the NOFDM used to calculate the values depends on the modulation.

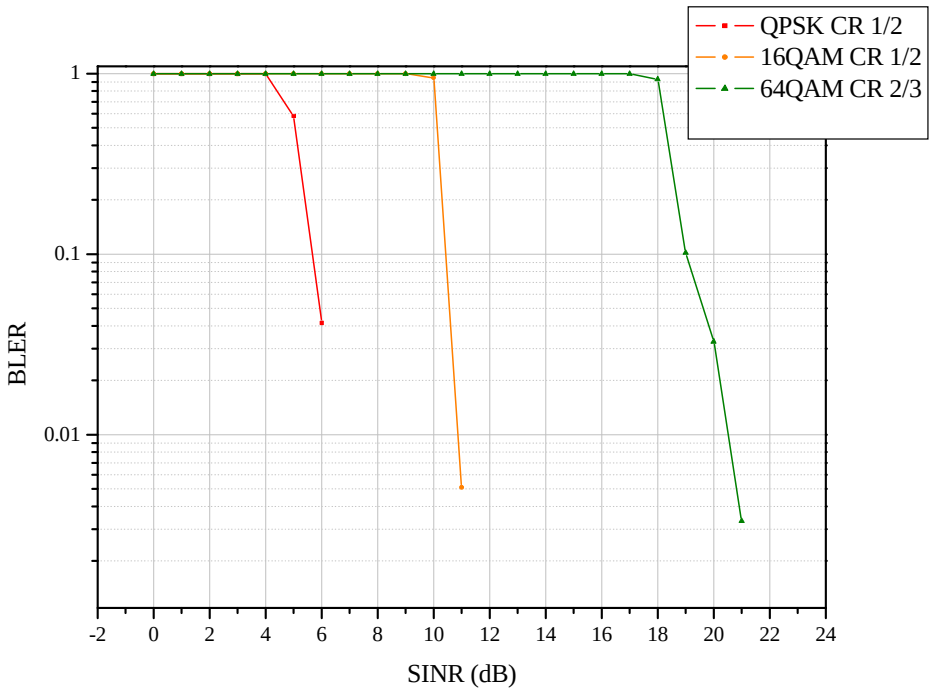

Figure 26 *SINR to BLER relation for different MCS. BL=1024 bits. Values from WiMAC tables*

Next figure shows the BLER value as a function of SINR and MCS achieved using OpenWNS tables. The BL used is 1024.

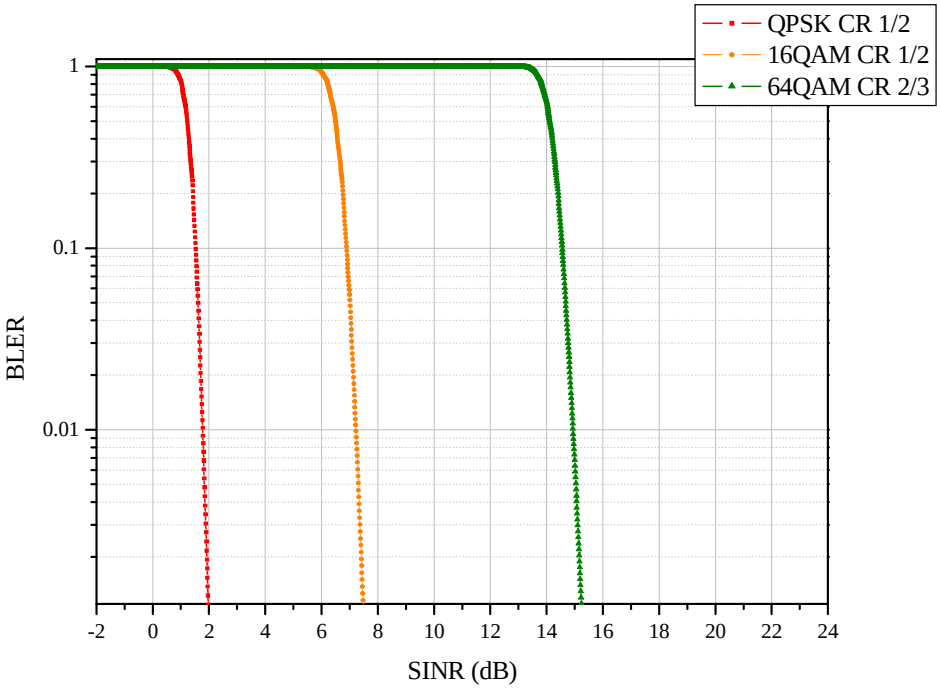

Figure 27 *SINR to BLER relation for different MCS. BL=1024 bits. Values from OpenWNS tables.*

Next table shows some of the values used to print the graphics above. Note that the SINR point taken as a reference is again when the BLER is approximately 0.1.

Table 8 SINR values when BLER=0.4 for different MCS and BL=1024 bits , using values from WiMAC and OpenWNS. Comparative

	SINR WiMAC	SINR OpenWNS	Increment/Decrement
QPSK CR 1/2	5.6 dB	1.5 dB	4.1 dB
16QAM CR 1/2	10.4 dB	7 dB	3.4 dB
64QAM CR 2/3	19 dB	14.5 dB	4.5 dB

By comparing Table 7 and Table 8, it can be seen that the difference has increased from 2.6 dB to 4 dB on average when the BL simulated goes from 256 bits to 1024 bits. The reason for this is double.

First, WiMAC approach to calculate the BLER gets worse when using longer BL, due to the use of(4.6). Consequently, a higher SINR is required to achieve the same BLER.

Second, due to the coding scheme used in each simulator. WiMAC used a combination of Reed-Solomon and Convolutional Codes [17]. OpenWNS, instead, includes the use of Turbo Coding technique, presented in 1993 [2], which outperform Reed Solomon and Convolutional Codes. It has been already said in 4.3.2.2.2 that Turbo Codes provide a better performance the longer block length used. Therefore, OpenWNS L2S interface improves its performance with the BL, finally providing a higher gain with respect of WiMAC approach.

CHAPTER 5

Evaluation of Hybrid ARQ schemes

This chapter analyzes the expected performance provided by the H-ARQ schemes, among other interesting concepts related.

5.1. LTE Physical Layer

Section 3.4.2 briefly described the H-ARQ protocol designed to be present on LTE technology. As it was mentioned, H-ARQ spans both MAC and PHY, being the last one the layer which will perform the combining process.

Figure 28 depicts a simplified PHY and MAC processing for the Downlink Shared Channel, DL-SCH.

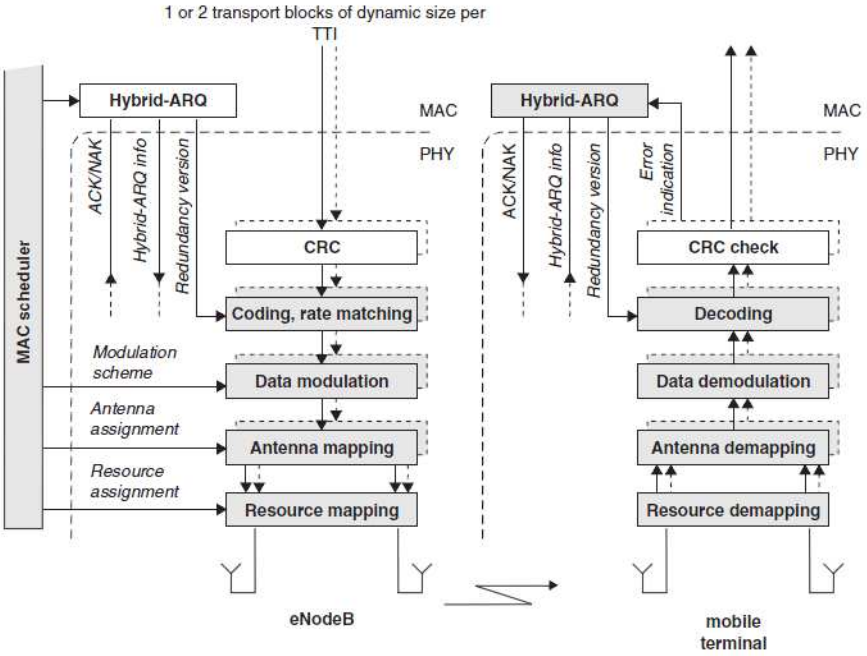

Figure 28 *PHY processing for DL-SCH [12]*

As it can be seen, the PHY controls the coding, the PHY H-ARQ processing, the modulation, the antenna mapping in case MIMO systems are used, and allocates the data to the suitable frequency-time resource. The decisions regarding H-ARQ protocols are taken into the MAC layer.

Most of those tasks are already covered by OpenWNS functionalities, and the rest are being studied, like the use of MIMO schemes. However, the important issue concerning this thesis is the fact that H-ARQ protocols directly or indirectly affect most of those functionalities. Within those, the scheduler would present the bigger complications if modifications to include H-ARQ mechanisms want to be done.

5.3 includes some protocol advices to be follow in order to include H-ARQ protocols within system level simulators, like OpenWNS. However, up to know, the whole implementation is considered out of this thesis scope.

Focusing again in Figure 28 , it can be seen that 1 or 2 transport blocks, depending on whether spatial multiplexing is used, are received by the PHY in a Transmission Time Interval (TTI). Some Cyclic Redundancy Check (CRC) bits are attached to the blocks. Those blocks are then coded. The downlink scheduler selects the CR and modulation scheme to be used in each block, depending on the block length and the amount of resources allocated for each transmission. Within this process, the H-ARQ protocols informs about the redundancy version to be used, affecting the coding process just explained. There it is one of the implications mentioned above that would complicate a lot the implementation of H-ARQ protocols at a different time when the scheduler is revised. Continuing with the downlink processing, the scheduler controls the antenna mapping in case spatial multiplexing had been used.

On the other side of the transmission, the scheduled mobile receives the signal and performs the reverse processing actions taken at the Base Station. The PHY informs the H-ARQ protocol whether the decoding has been successful or not. The MAC part of the H-ARQ decides then whether a retransmission is required.

The PHY processing for the Uplink SCH is similar to the DL explained above. Further details can be seen in [12].

5.2. Evaluation of H-ARQ Protocols

As it was mentioned, H-ARQ protocols are still not present in OpenWNS. In order to avoid a lot of restrictions and reach a good an accurate performance, it is important than the whole protocol stack, with special attention to the scheduler, is implemented or revised at the same time H-ARQ is included.

Despite of this, the expected performance that OpenWNS would provide when using H-ARQ together with the L2S interface analyzed along this thesis can be evaluated. In order to do this, a series of tests have been implemented, providing a comparison performance between CC and IR schemes. That comparison will be presented in terms of predicted BLER and Throughput achieved.

Note that it is interesting to see how the H-ARQ protocols are going to use the L2S interface analyzed along this thesis.

5.2.1 Chase Combining

A theoretical study of CC scheme can be found in 3.2.2. This section describes the PHY abstraction necessary to model the Chase Combining protocols, working together with the L2S interface and its mapping mechanisms. After that, a performance evaluation is given.

5.2.1.1 PHY Abstraction for Chase Combining

The SINR values for the different transmissions of the same block are summed. The resulting SINR must be treated according to the mappings implemented in each case. If MIESM is the mapping implemented, the resulting SINR is used to calculate a MI value following the next formula [6]:

$$MI = \frac{1}{N}\sum_{n=1}^{N} I_m\left(\sum_{j=1}^{q} SINR_{nj}\right) \tag{5.1}$$

Where q is the number of transmissions, I_m is the MI function for the modulation order m and $SINR_{nj}$ is the n-th symbol SINR during j-th retransmission. In the special case of OpenWNS, one block has one SINR, and not one SINR per symbol. Therefore, formula above can be reduced and seen as:

$$MI = I_m\left(\sum_{j=1}^{q} SINR_j\right) \tag{5.2}$$

In case EESM had been implemented, the effective SINR would have been given by:

$$SINR_{eff} = -\beta \ln\left(\frac{1}{N}\sum_{n=1}^{N}\exp\left(-\frac{\sum_{j=1}^{q} SINR_{nj}}{\beta}\right)\right) \tag{5.3}$$

Where $SINR_{eff}$ is the effective SINR after q transmissions.

5.2.1.2 Simulation Scenario for Chase Combining

Different assumptions were taken in order to facilitate the performance evaluation of CC schemes.

First, it is assumed that the MCS used for the retransmissions is the same as the one used for the initial transmission. According to this, (5.2) can be seen now as:

$$MI = I\left(\sum_{j=1}^{q} SINR_j\right) \tag{5.4}$$

Where I represents now the SINR to MI mapping for a specific modulation scheme, which will remain the same along the retransmissions. Note that the SINR is added in

linear units, and then the SINR in dB is calculated before convert it into a MI by using the MIESM.

Finally, it is assumed that the retransmissions are received with the same energy as the initial transmission. Consequently, (5.4) can be seen now as:

$$MI = I\left(\sum_{j=1}^{q} SINR_j\right) = I\left(SINR \cdot \sum_{j=1}^{q} 1\right) = I\left(q \cdot SINR\right) \quad (5.5)$$

The initial transmission is received then with SINR dB. After one retransmission, the effective SINR is double the initial SINR in linear units or the initial SINR + 3dB. After two retransmissions, initial power multiply per three or SINR + 4.7 dB and so on.

Once the MI value is calculated using the MIESM, studied in 2.2.1.3 and 4.3.1, a BLER can be obtained using the MI to BLER mapping analyzed in 2.2.2 and 4.3.2. The modulation, acting as a parameter for the first mapping, does not change after the retransmissions. The code rate and the block length, used within the second mapping, do not change either, as it was presented in 3.2.2.

The BLER obtained is used to decide whether to ask for a retransmission. In order to do that, the BLER achieved after the initial transmission or consequent retransmissions is compared with a random value with a uniform distribution between 0 and 1.

A maximum number of retransmissions allowed is set within the tests.

5.2.1.3 Performance Evaluation

As it was mentioned, the performance evaluation will be given in terms of BLER and Throughput achieved for different SINR.

5.2.1.3.1 BLER

Since retransmissions are now allowed, and even more, they are being combined to create a more reliable codeword, the expected BLER compared with the BLER achieved when not H-ARQ were used is lower given the same SINR.

Moreover, the BLER decreases when the allowed number of retransmissions increases.

Figure 29 depicts the SINR to BLER relation for two MCS, comparing the performance achieved when no H-ARQ is used with the performance achieved when CC scheme is used. The lines for different number of retransmissions are depicted.

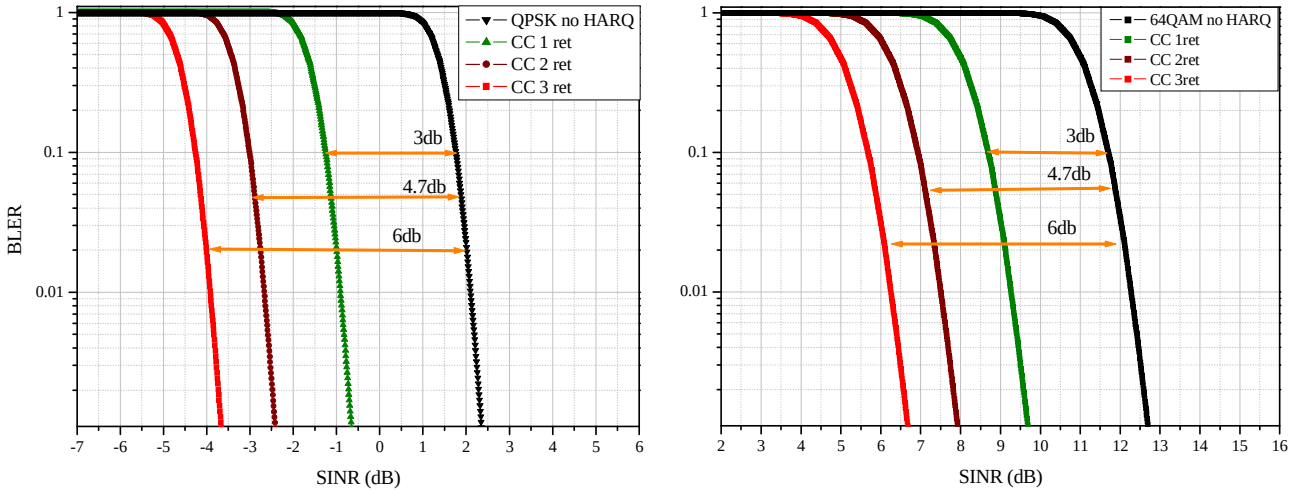

Figure 29 *SINR to BLER relation for QPSK and 64QAM, allowing up to 3 CC retranmissions. CR 1/2, BL 512 bits*

It can be seen how the BLER curve resulting after each retransmission is the same as the original one moved to the left a certain dB value, following the results provided by (5.5). The same behaviour applies for each MCS possible.

5.2.1.3.2 Throughput

When it comes to throughput, it is important to clarify one aspect. The use of ARQ protocols or H-ARQ protocols causes a decrease in terms of total throughput, since retransmissions are being transmitted instead of new information. Despite of this, the use of these protocols is completely necessary in order to provide reliable communications, where lost packets must be able to be retransmitted again. Moreover, what ARQ and now H-ARQ allows is the possibility of sending information at lower SINR than when no retransmissions were allowed.

Figure 30 depicts the Throughput achieve for a 64QAM with code rate 5/6 and block length of 512 bits. Results when using CC with up to 5 retransmissions allowed are compared with the results when not using H-ARQ. As it can be seen, the use of CC provides a throughput closer to Shannon, the limit of maximum transfer of information through a noisy channel.

The difference steps that can be appreciated represent the SINR for which values the transmissions will need one less retransmission on average (going from the left side to the right side). Consequently, the throughput increases. The values for the different steps that can be seen represent the maximum possible transfer for that MCS divided per the number of transmissions required. The next formula is applied:

$$Throughput = \frac{\frac{bits}{symbol} \cdot CodeRate}{\#transmissions} \tag{5.6}$$

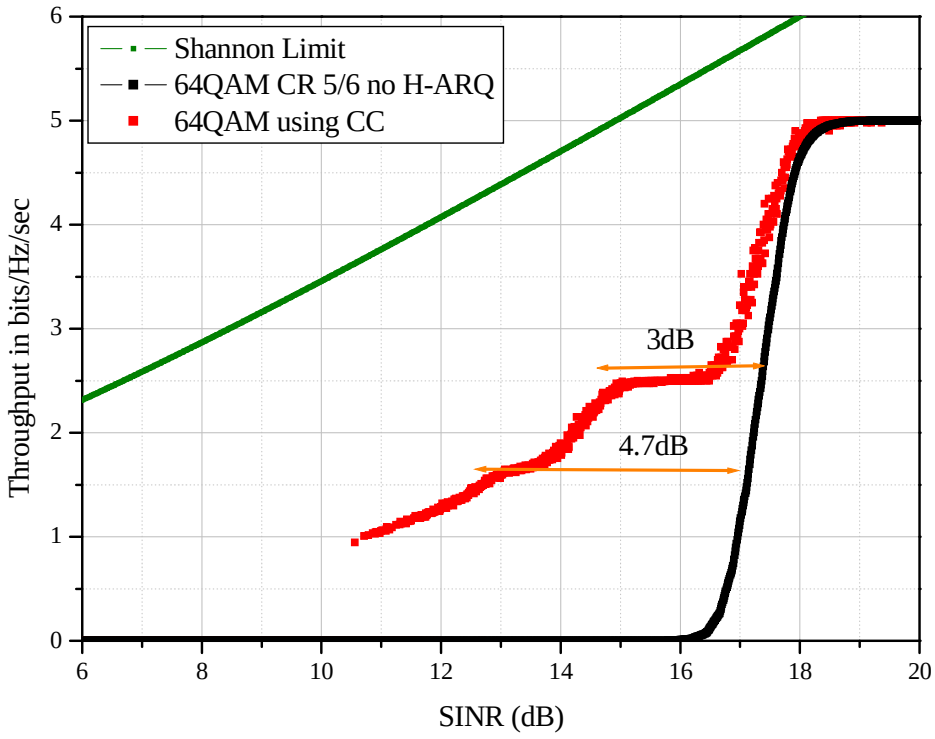

Figure 30 *Througput for 64QAM CR 5/6 , BL 512, using or not CC, allowing up to 5 transmissions*

Please note how it is now possible to send information at lower SINR. The different steps dB differences correspond to the difference in terms of BLER seen at Figure 29 .

5.2.1.3.3 Modified Scenario

A little modification has been made for this second scenario, trying to evaluate how accurate is one of the assumptions taken up to now. Specifically, the received SINR for the retransmissions considered within this second scenario might not be the same as the initial transmission. However, it is assumed that the new value will be around the initial one. In order to simulate that, the predicted SINR for a retransmissions is calculated by multiplying the initial transmission SINR power per a random value with uniform distribution. Different limits are considered. From 0.8 to 1.2, so considering the retransmissions will be received with ±20% of the initial transmission energy. From 0.7 to 1.3, so ±30%, and finally from 0.6 to 1.4, so ±40% of the initial energy.

Next formula is used then in order to calculate the effective MI.

$$MI = I_m\left(initialSINR + \sum_{j=1}^{r} \alpha \cdot initialSINR\right) = I_m\left(initialSINR \cdot (1 + \sum_{j=1}^{r} \alpha_j)\right) \quad (5.7)$$

Where *initialSINR* is expressed in linear values, r is the number of retransmissions, and α_j is the random value for each one of the retransmissions.

The results obtained by modelling this second scenario must be really close to the ones achieved for the first scenario. The reason is that a random value with uniform distribution between 0.7 and 1.3, 0.8 and 1.2 or 0.6 and 1.4 has an average value of 1, which means that, looking at (5.7), the received SINR for the retransmissions is finally the same as the initial transmission, as it happened in the first scenario.

Figure 31 depicts the differences in terms of throughput for a specific MCS when assuming retransmissions are received with same energy as initial transmission or received with a ±% of the initial transmission energy.

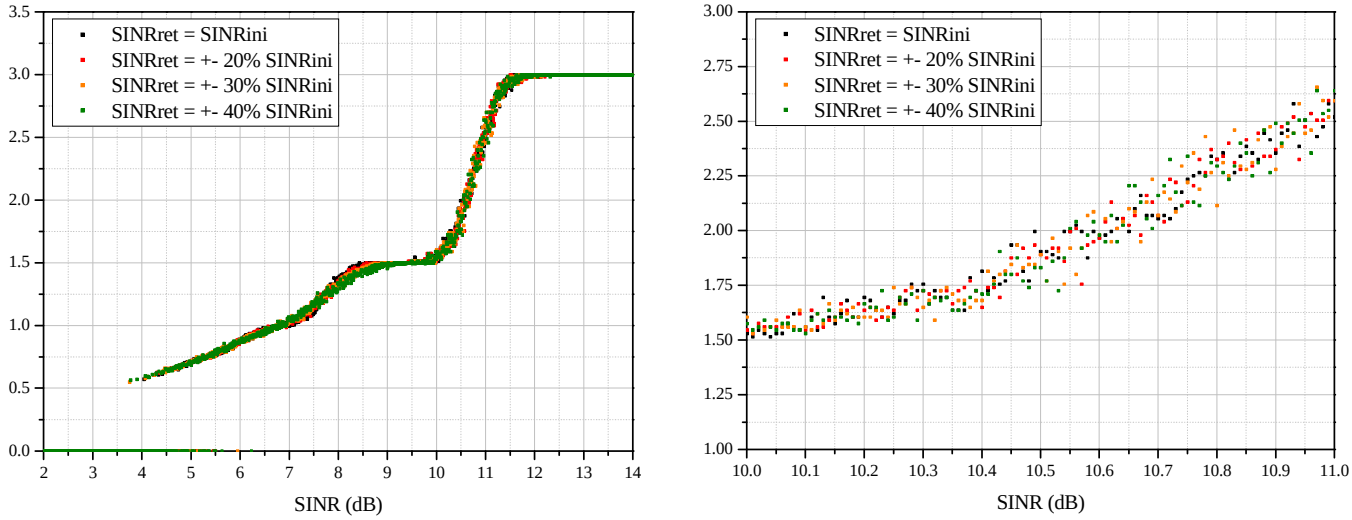

Figure 31 *Comparison between throughput achieved by using first or second scenario. 64QAM CR 1/2 BL 1024 bits. Zoom in right figure*

As it was predicted, differences are minimal. The interesting thing here is then to study the standard deviation for each one of the percentages used. The formula used is:

$$\sigma_{\%} = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(Thro_i - \overline{Thro})^2} \qquad (5.8)$$

Where $\sigma_{\%}$ is the standard deviation for each one of the ±% considered, N is the number of values taken into account, $Thro_i$ is the throughput achieved when using the second scenario, and $\overline{Thro}$ is the average throughput, obtained by using the first scenario scheme.

Throughput values obtained for SINR from 4 to 12 dB are taken into account, obtaining the following results.

Table 9 Standard deviation in terms of throughput for different percentages assuming second scenario.

	20% initial energy	30% initial energy	40% initial energy
Standard deviation σ	0.17880 bits/Hz/sec	0.18054 bits/Hz/sec	0.18358 bits/Hz/sec

As it was expected, the standard deviation increases with the % considered. However, differences for any of the % considered are small enough to conclude that the first scenario, assuming same received SINR for initial transmissions and retransmission is accurate enough.

This thesis will not consider any other scenario, like accepting any possible SINR value for a retransmission not related with the initial received SINR.

5.2.2 Incremental Redundancy

The theoretical study, interesting concepts and different proposals for IR H-ARQ schemes can be found in 3.2.3. This section, as the section above, will analyze the PHY abstraction needed to evaluate the performance of the IR protocol. After that, a performance evaluation is given.

5.2.2.1 PHY Abstraction

The PHY abstraction when using Incremental Redundancy schemes depends on whether retransmissions with repeated coded bits are allowed or not.

5.2.2.1.1 No Repeated Coded Bits Allowed Scenario

If no repeated coded bits are allowed, new redundant bits will be sent in each retransmission, and no repeated coded bits will be included. That leads us to a code word that looks like this:

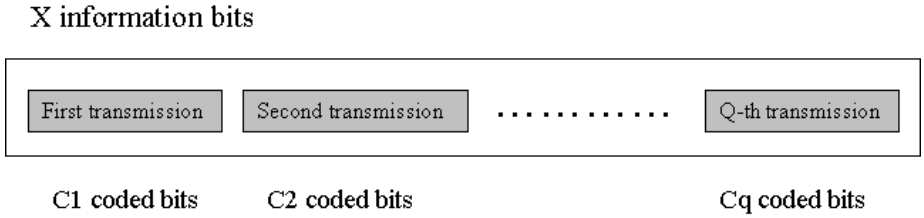

Figure 32 *Example of a codeword where no bits will be sent more than once*

Where X is the set of information bits, q the number of transmissions, and C_i the number of coded bits sent into the $i-th$ transmission.

As it was mentioned in 3.2.3, the code rate and the block length after each IR retransmission change, differently from CC schemes.

Therefore, the inputs for the L2S interface mappings regarding this particular scenario follow next formulas, from [6]:

$$effectiveCR = \frac{X}{\sum_{i=1}^{q} C_i} \tag{5.9}$$

$$effectiveBL = \sum_{i=1}^{q} C_i \tag{5.10}$$

$$effectiveMI = \frac{\sum_{i=1}^{q} C_i . MI_i}{\sum_{i=1}^{q} C_i} \tag{5.11}$$

Where *effectiveCR* , *effectiveBL* and *effectiveMI* are the effective CR, BL and MI after q transmissions, and they are the values that will act as an inputs of the L2S mappings.

5.2.2.1.2 Repeated Coded Bits Allowed Scenario

The first case studied above represents a single and restricted scenario. Normally, partial repetition of coded bits is possible, and that need to be taking into account for the PHY abstraction.

The formulas used then to compute the MI, the BL and the CR after a retransmission are:

$$effectiveMI = \frac{N_{pre} \cdot MI_{old} + N_{NR} \cdot I_b + N_R \cdot f_1\left(f_1^{-1}\left(MI_{old}\right) + f_1^{-1}\left(I_b\right)\right)}{N_{pre} + N_{NR} + N_R} \tag{5.12}$$

$$effectiveCR = \frac{X}{N_{pre} + N_{NR} + N_R} \tag{5.13}$$

$$effectiveBL = N_{pre} + N_{NR} + N_R \tag{5.14}$$

Where N_{NR} is the set of bits not repeated, new, N_R represents the set of bits repeated from previous transmissions, N_{pre} represents the set of bits not retransmitted in the last attempt, but retransmitted before, I_b is the averaged mutual information per bit for the last transmission, and $f_1(\ )$ is the mapping from SINR to MI. In case the modulation used is the same for every transmission, the $f_1(\ )$ must be the mapping corresponding to this modulation. In case the modulation scheme can change along the retransmissions, it should be use the $f_1(\ )$ corresponding to QPSK [6].

If repeated coded bits are allowed, the IR retransmissions not only would provide a coding gain, since new bits are transmitted, but an accumulated energy gain too, because some bits will be transmitted more than once.

5.2.2.1.3 Intermediate Scenario

An intermediate scenario, not so reduced as the one presented in 5.2.2.1.1, but not so complicated as the one presented above, would be to allow repeated coded bits into the retransmissions, but not before all the redundant bits from the initial codeword are exhausted. This is the scheme that will be modelled and evaluated from now on, and it is similar to the one presented in 3.2.3.2.

5.2.2.2 Simulation Scenario

The simulation scenario corresponds to the scenario presented in 5.2.2.1.3. However, it has been modified due to a peculiar characteristic of the Turbo Codes, used within OpenWNS. Further details about Turbo Codes can be seen in Appendix D, but the main point is the following. If the coding gain (e.g. in Eb/No) achieved by using Turbo Codes for different code rates is plot, it can be seen that a big gain is achieved when the code rate starts to decrease from CR=1. However, this additional coding gain becomes smaller the lower the CR gets. It has been studied that a CR=1/3 marks somehow the limit where the coding gain achieved starts to remain almost constant even thought the CR decreases even more. Moreover, it has been demonstrated that the performance provided by higher code rates than 1/3 (e.g. 1/2) created from puncturing an initial codeword with CR=1/3, is almost the same as the performance provided by codeword directly coded with the higher CR.

Due to this, no lower CR that 1/3 are used within Turbo Codes, and higher CR are created by puncturing an initial codeword with CR=1/3.

Let's see how come this characteristic will determine the model implemented to test the IR scheme.

As it has been explained in section above, the effective CR and the BL that will feed the L2S mappings change after each IR retransmission. The received energy per bit must be actualized too in case repeated bits are transmitted.

Some assumptions are taken in order to facilitate the evaluation. First, it is assumed that the modulation coding scheme does not change along the retransmissions. Every transmission of the same block will be coded and modulated equally.

Second. Even though it was said that the BL must be actualized following formula (5.10) or (5.14), at the end this block will arrive to a Turbo decoder (at least in OpenWNS case), that will have a fixed number of inputs. If fewer bits arrived, some of the entries will be set to zero, and thus the resulting bits currently used will be always the same.

Finally, it is important to remember one more time that no CR lower than 1/3 will be considered. Moreover, no repeated bits are allowed until all the bits from the initial codeword have been sent. With these, the effective CR and SINR which will feed the mappings follow next formulas:

$$effectiveCR = \max\left(\frac{X_{payload}}{N_{CodedBitsTransmitted}}, \frac{1}{3}\right) \tag{5.15}$$

$$effectiveSINR = initialSINR \cdot \left(1 + \frac{N_R}{C_{total}}\right) \quad (5.16)$$

Where N_R is the set of total bits repeated, independently of in which transmission, N_{NR} is the set of bits sent just once, and C_{total} is the total number of coded bits, which in Turbo Coding is always $C_{total} = 3 \cdot X_{payload}$, due to the turbo decoder characteristic explained above.

Let's exemplify this for a specific CR in order to clarify the ideas. Figure 33 depicts the process and how the variables are refreshed. Note that the puncturing is used in order to achieve a CR=1/2 for each one of the redundancy versions generated from the initial codeword with CR=1/3. Note that no repeated coded bits are retransmitted before all the bits from the initial codeword have been sent.

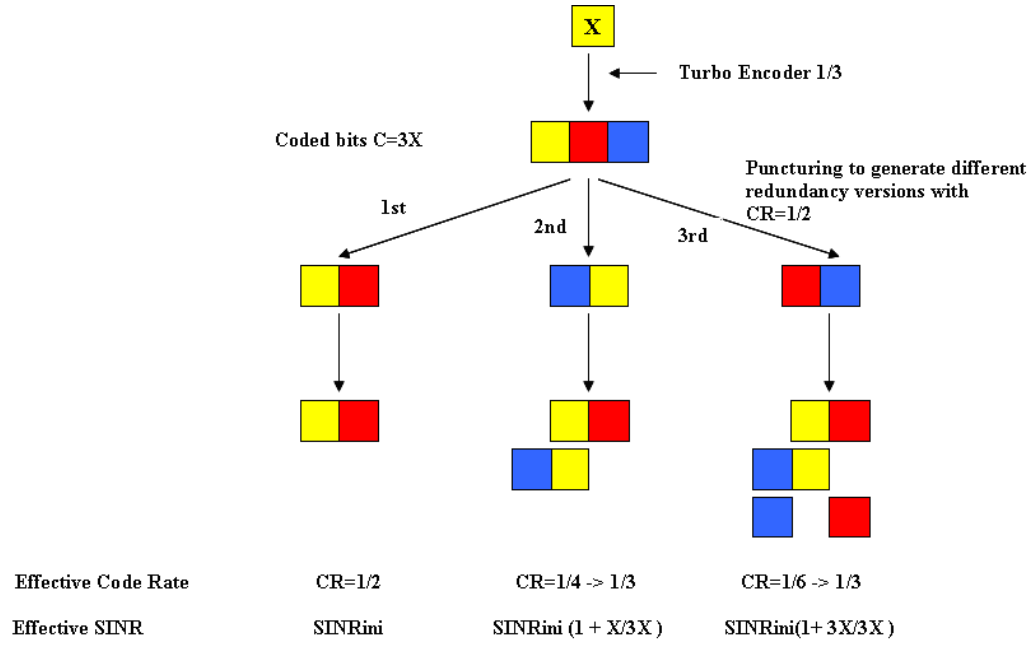

Figure 33 *Model of the scenario implemented*

Regarding the code rate, it can be seen that the code rate for the initial transmission, higher than 1/3, will feed the mappings, following (5.15). However, the effective code rate after the first retransmission would be CR=1/4, but the model, following the characteristics of Turbo Codes explained above, will take CR=1/3 (without losing a considerable coding gain due to the assumption). Same thing happens after the second transmission. Regarding the effective SINR, it follows (5.16).

A similar scenario to the one in CC has been implemented to test the IR scheme. The different BLER obtained after each retransmission are compared to a random uniform value between 0 and 1, deciding after that whether to retransmit or not. In case a retransmission is required, the effective SINR and CR are refreshed following (5.15) and (5.16), and feed again the L2S mappings.

5.2.2.3 Performance Evaluation

Again, the performance evaluation is given in terms of BLER and Throughput.

5.2.2.3.1 BLER

As it happened with CC, a gain in terms of BLER is expected when using IR with respect of not using any H-ARQ scheme.

Next figures depict the SINR to BLER relation for three different MCS, showing the difference performance obtained when using IR with respect of not using any H-ARQ technique.

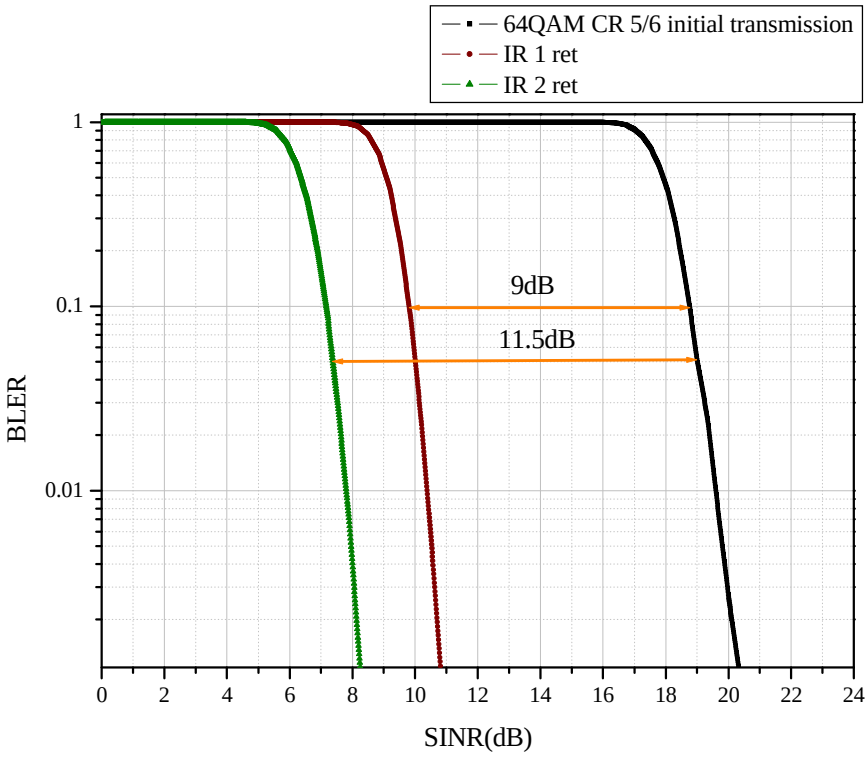

Figure 34 *SINR to BLER relation for 64QAM, allowing up to 2 IR retranmissions. CR 5/6, BL 512 bits*

Let's explain the meaning of the lines that can be seen in figure above. Even though the example was given for CR 1/2, it can be helpful to look again at Figure 33 to clarify the ideas that will be given in here.

There is no secret for the initial transmission. Tables from OpenWNS have been used to depict the SINR to BLER relation for this MCS. The initial transmission sends $6/5 \cdot X_{payload}$ bits. The interesting points start with the retransmissions. After a first IR retransmission, the decoder has the $6/5 \cdot X_{payload}$ bits from the initial transmission plus $6/5 \cdot X_{payload}$ bits from the retransmission. That gives a total of

$12/5 \cdot X_{payload} = 2,4 \cdot X_{payload}$ from the initial $3 \cdot X_{payload}$ generated by the Turbo Encoder 1/3. Therefore, there are no repeated coded bits up to now, so there is no energy accumulated. The effective CR is 5/12, higher than 1/3, so it feeds the mappings (5.15). The gain of 9dB that can be seen is therefore due to a coding gain. After the second retransmission, a total of $3,6 \cdot X_{payload}$ bits have been sent from the initial $3 \cdot X_{payload}$. Therefore, $0,6 \cdot X_{payload}$ bits have been repeated, and the accumulated energy must be actualized following (5.16). The effective CR would be 5/18, but the model will take CR 1/3, following (5.15). In that case, the gain is not only provided by the coding gain, but by the accumulated energy too.

Let's see now the performance for an initial CR 1/2.

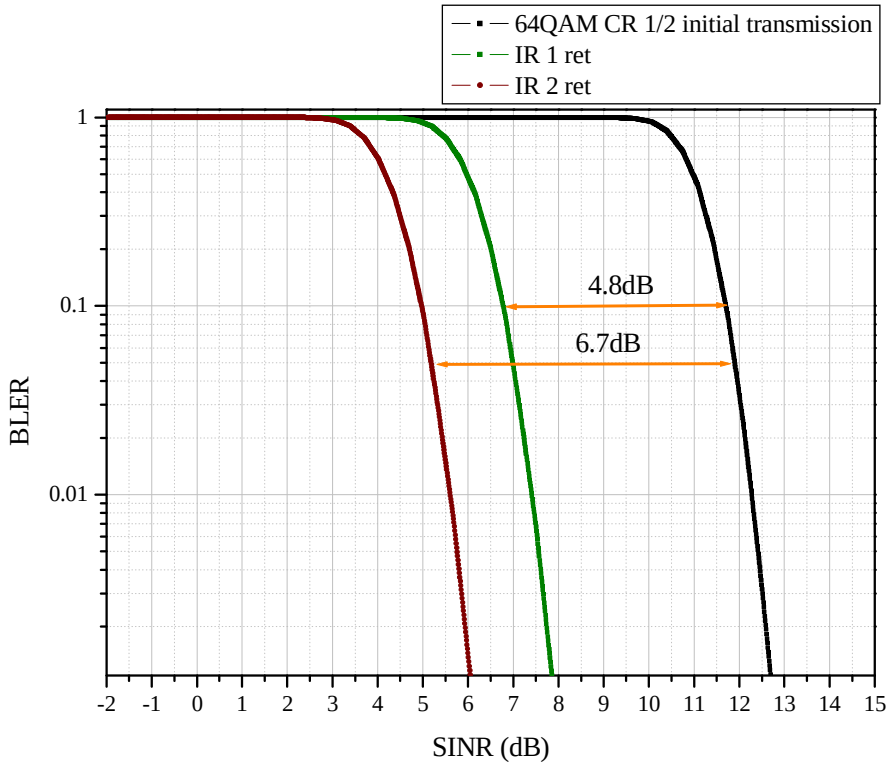

Figure 35 *SINR to BLER relation for 64QAM, allowing up to 2 IR retranmissions. CR 1/2, BL 512 bits*

The same study is shown in here. After the first retransmission, the decoder has $4 \cdot X_{payload}$ bits from the initial $3 \cdot X_{payload}$. Therefore, $X_{payload}$ bits have been sent twice. The energy accumulated then must be taken into account. Regarding the CR, the effective one would be 1/4, but the model will use 1/3. After the second retransmission, a total of $6 \cdot X_{payload}$ bits have been sent, so $3 \cdot X_{payload}$ are repeated, the total codeword. The energy would be now double, and the effective CR 1/3, even though it would be really 1/6.

The last example, quite interesting, is given for CR 1/3. Note that in that case, there is no coding gain for any of the possible IR retransmission, since the whole codeword is sent every time.

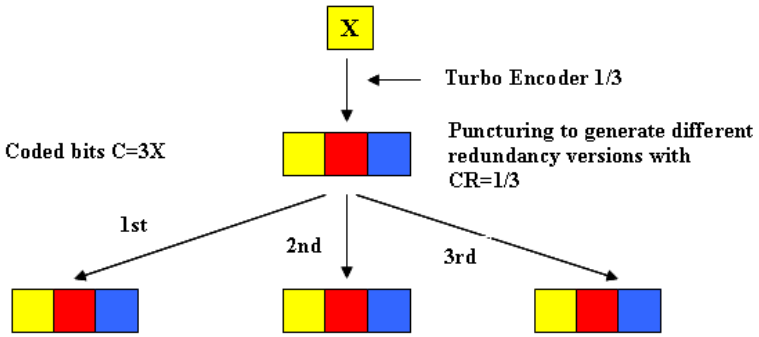

Figure 36 *Example of redundancy version sent for IR retransmission with CR 1/3*

Due to this, the expected performance is the same as in the Chase Combining scheme. The gain is only provided by the fact that the energy accumulated increases after each retransmission.

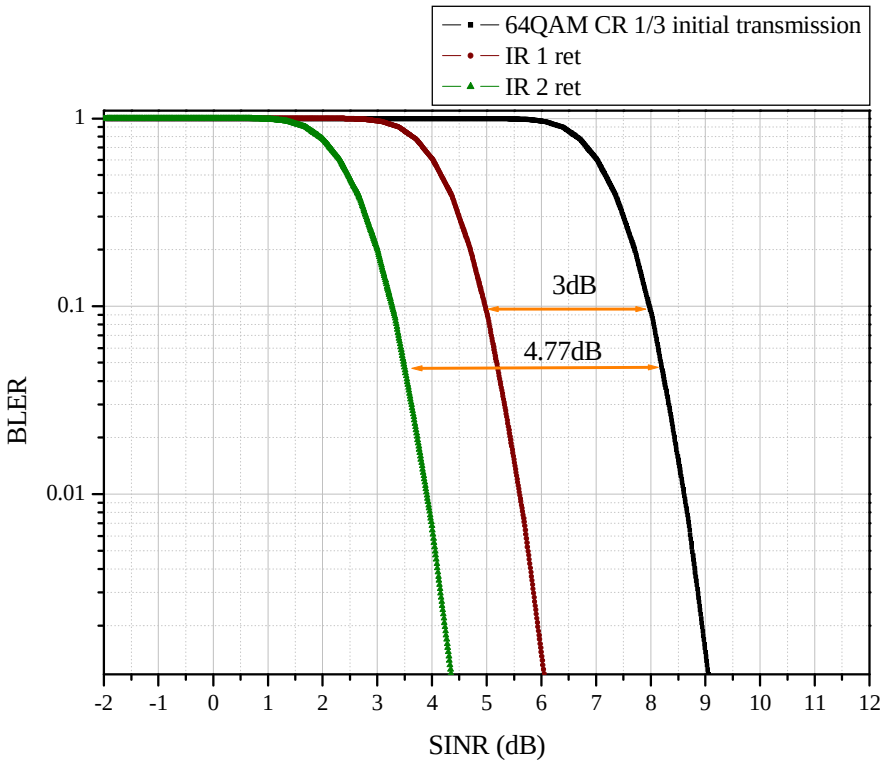

Figure 37 *SINR to BLER relation for 64QAM, allowing up to 2 IR retranmissions. CR 1/3, BL 512 bits*

Summing up, it has been seen than the gain provided by IR retransmission strongly depends on the coding scheme used, differently from CC, when the behaviour was the same independently of the MCS.

5.2.2.3.2 Throughput

Same considerations given in 5.2.1.3.2 about the throughput apply for the IR scheme case.

The throughput achieved when using IR retransmissions compared to the one achieved when no H-ARQ is used can be seen in Figure 38 .

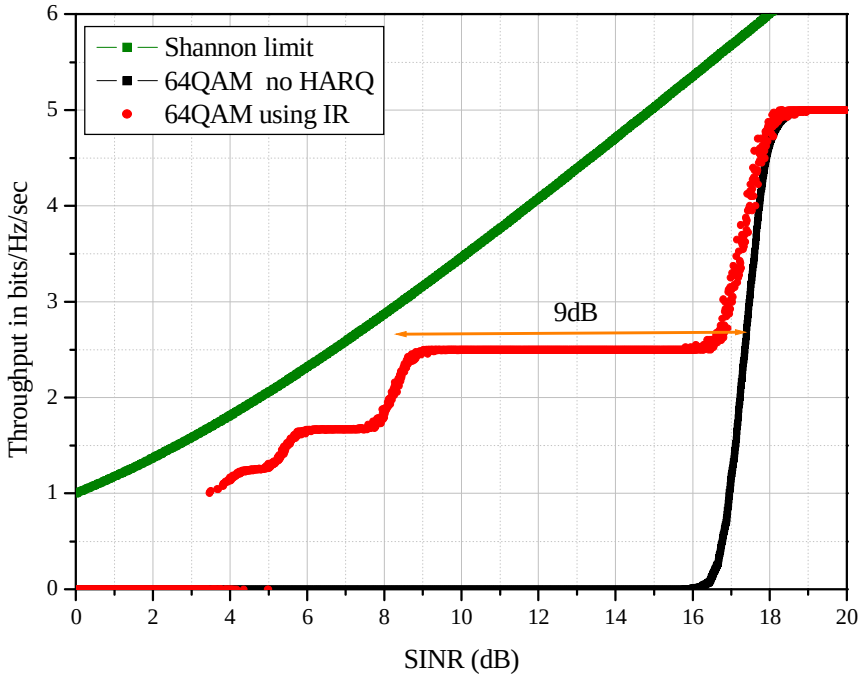

Figure 38 *Througput for 64QAM CR 5/6 , BL 512, using or not IR, allowing up to 5 transmissions*

As it can be appreciated, the throughput achieved when using IR is closer to the Shannon limit. Note how the width of the step in dB corresponds to the gain seen in Figure 34 for the first retransmission.

5.2.3 H-ARQ Schemes Comparison

Once both H-ARQ schemes performance have been presented, it is time to compare them. Before that, thought, it can be interesting to remember again why CC and IR provide a gain in terms of BLER or throughput, and which aspects differentiate each other.

CC provides a gain due to the accumulated received energy per bit. Every time a retransmission is required and received, the accumulated energy increases, thus the chances of successfully decoding the block.

Contrary, IR scheme provides a gain based on coding gain, since new bits might be transmitted in each retransmission, and based on an accumulated received energy per bit too, in the case of repeated bits are allowed.

Due to these characteristics, the differences between CC and IR do not remain the same, but depend on the coding scheme used. CC performance does not care about the MCS. The gain is always 10.log (#transmissions) dB if it assumed that the SINR for the retransmissions is the same as the SINR for the initial transmission. However, the gain provided by IR does depend on the coding scheme, and it is bigger the higher the initial CR used, as it has been shown in 5.2.2.3. Let's explain this.

As it was said in 3.2.3.1, the different redundant versions are created by puncturing the initial mother code. If the initial CR is high, the difference between this CR and the effective CR after a retransmission is received is bigger than if the initial mother code would have been coded with a low CR. It is important to remark again that no CR lower than 1/3 are used within Turbo Codes.

Figure 39 exemplifies where the gain comes from in IR retransmissions depending on the initial CR sent.

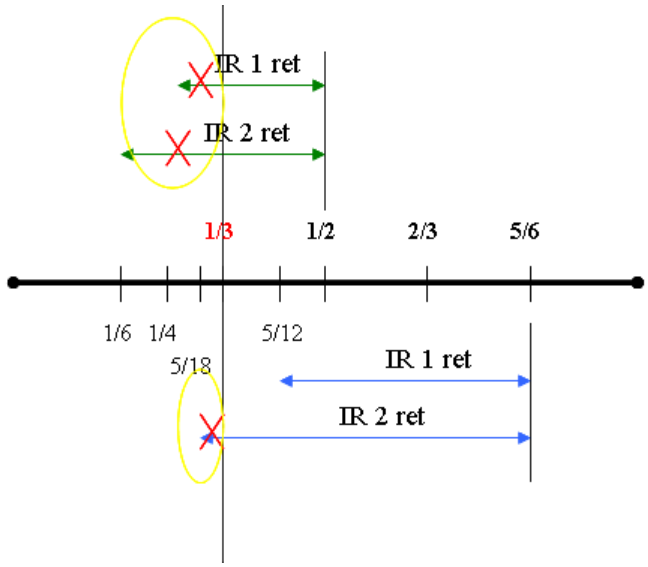

Figure 39 *Example of IR effectiveCR after retransmissions starting from CR 1/2 or 5/6*

When a puncturing to achieve a CR 5/6 is performed, and effective CR 5/12 is achieved after the first retransmission. That provides a big gain. If a second retransmission is required, the effective CR used within the model is 1/3, since not lower CRs are allowed. Anyways, there is an important coding gain too. Moreover, since there are repeated bits, as it was explained in 5.2.2.3, there is an accumulated energy per bit gain.

The gain provided when a puncturing to achieve a CR 1/2 is performed is lower. The reason is that CR 1/4 and 1/6 are not used within Turbo Codes, and even though they will, the gain would be almost the same as the one provided by CR 1/3. Therefore, it can be seen that the difference between 5/6 and 5/12 is bigger than the difference between 1/2 and 1/3. This is reflected in the SINR to BLER curves as it was studied in 4.3.2.2.2.

Next figures exemplify the SINR to BLER relation for a 64QAM modulation and different starting CR, as well as the performance achieved when using CC or IR, with

a maximum of 2 retransmissions allowed. The comparison study starts with the higher code rate available in OpenWNS, 5/6, to decrease then to a CR 1/2 and finally to 1/3.

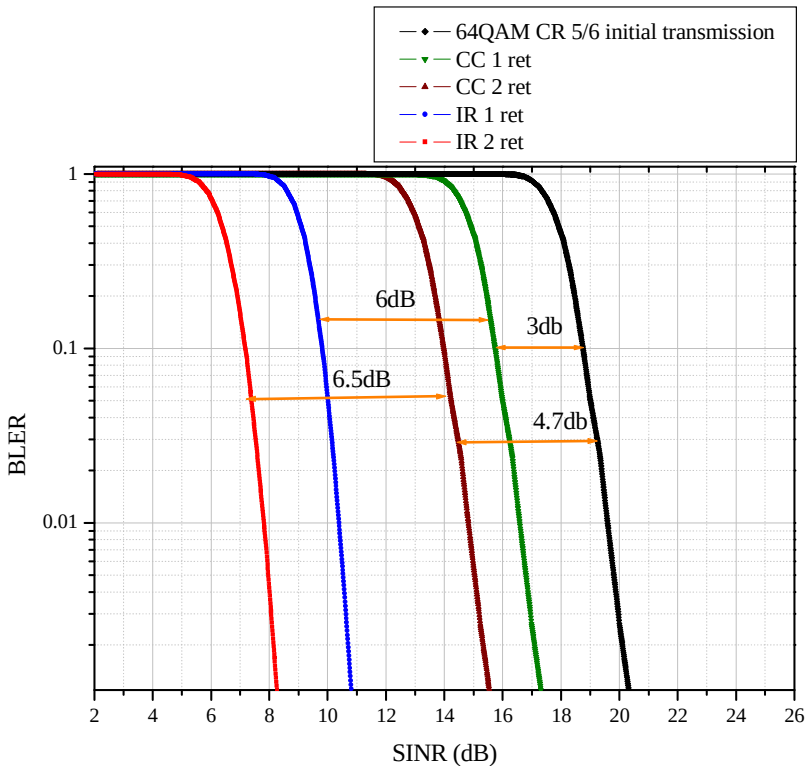

Figure 40 *SINR to BLER for 64QAM. Initial CR 5/6, BL 512 bits*

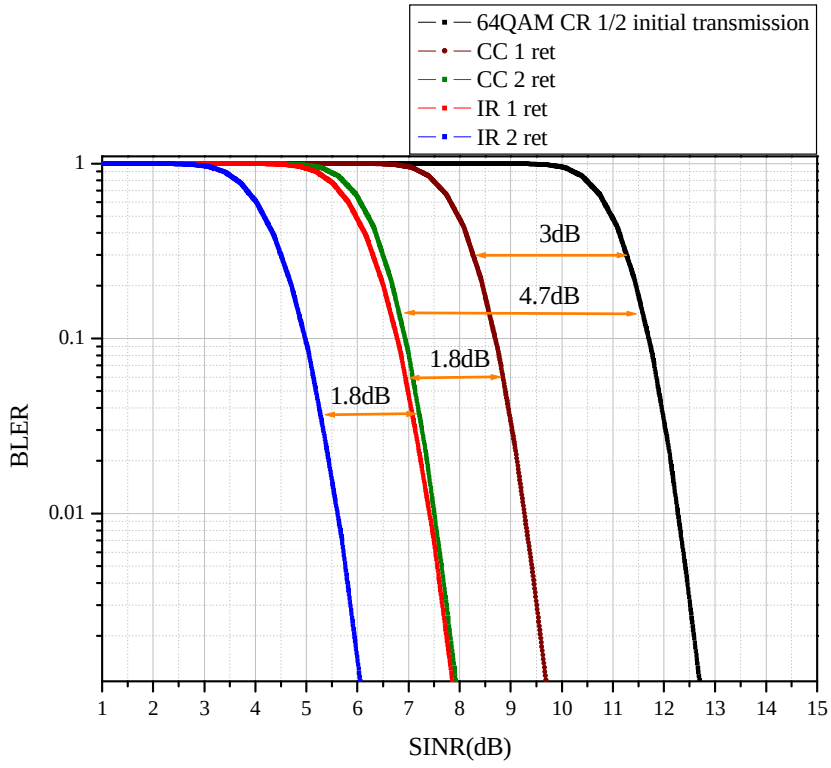

Figure 41 *SINR to BLER for 64QAM. Initial CR 1/2, BL 512 bits*

By looking at both figures it can be see how, as it was advanced, the gain using CC remain the same independently from the coding scheme used. However, the IR scheme reduces its gain around 4dB when starting from a CR 1/2 instead of CR 5/6. Therefore, the distances between both schemes have been reduced.

The last figure of this series shows the same curves, but when starting with a CR 1/3. As it was predicted in 5.2.2.3, the behaviour of both schemes is the same for this initial CR, since it is not possible to obtain any coding gain. All the gains provides then of the energy per bit accumulated along the retransmissions

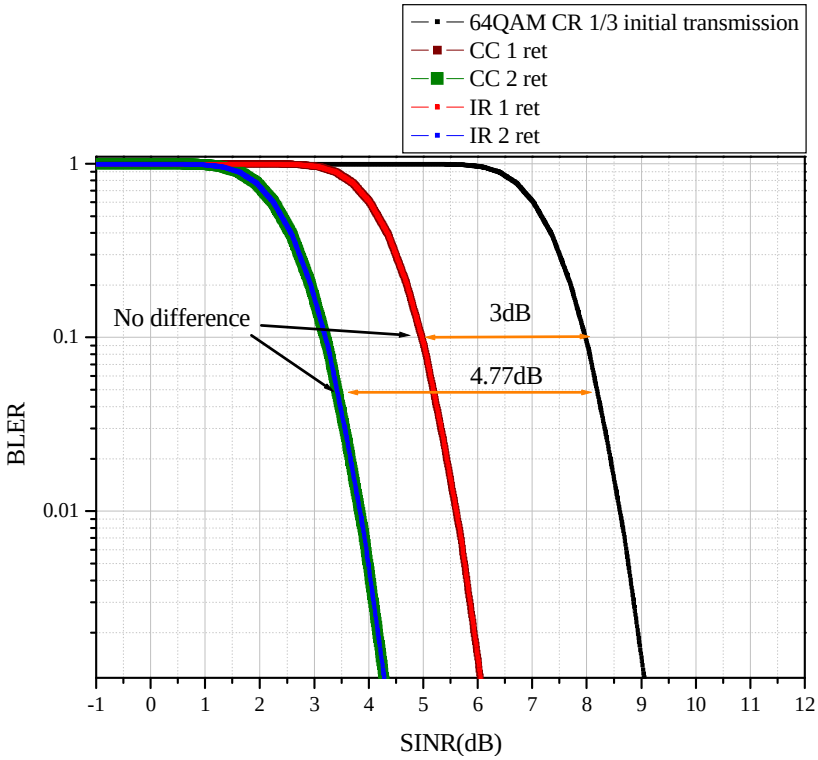

Figure 42 *SINR to BLER for 64QAM. Initial CR 1/3, BL 512*

There is a direct relation between the BLER and the expected throughput achieved. Therefore, it is not a surprise that the same behaviour seen for the BLER applies for the throughput. Next figure depicts this relation for 64QAM, starting with three different CR. This time up to 4 retransmissions are allowed.

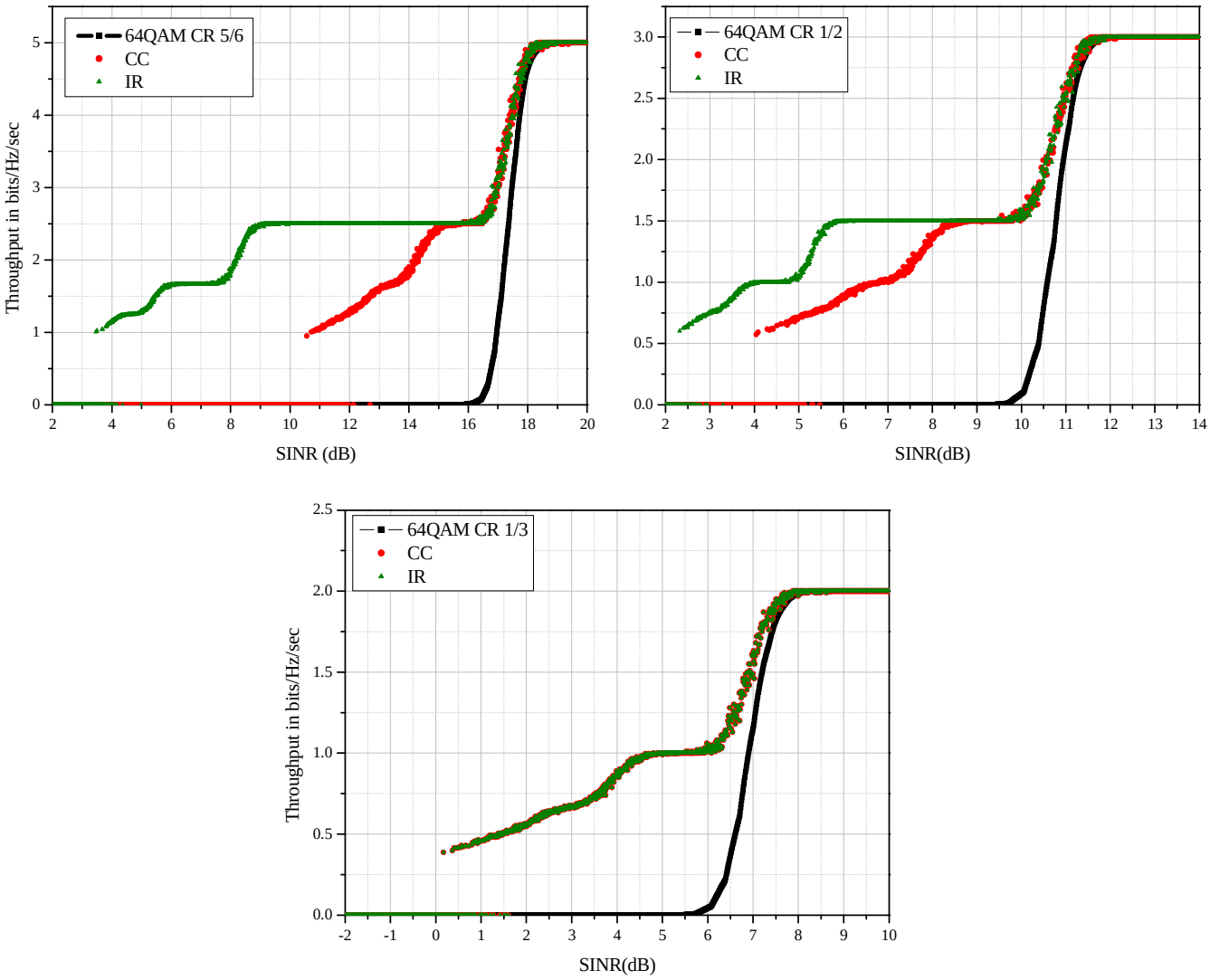

Figure 43 *Throughput achieved for 64QAM and 3 different initial CR. BL 512. Up to 4 tranmissions allowed*

As it was expected, the differences between CC and IR schemes decrease with the initial CR. The differences for the first step, corresponding to the first retransmission, are equal than the differences shown in Figure 40 , Figure 41 and Figure 42 for the BLER.

As a final state, it can be said that an intelligent decision would be to use Chase Combining scheme when the initial CR is low, close to 1/3, since the performance compared to IR is almost the same, as CC is easier to implement. IR should be used when the initial CR is high, since the coding gain that will be achieved with the retransmissions overcomes the energy per bit accumulated gain provided by CC.

5.3. Advices to Implement H-ARQ in OpenWNS

This section just expects to provide some simple advices in case H-ARQ protocols want to be introduced into system level simulator. Along the theoretical study of H-ARQ given in CHAPTER 3, and the performance evaluation given in CHAPTER 5, some details have been presented. Some of the ones missing, important to model the protocol, can be found in here.

Each user has a H-ARQ entity. Each one of these entities, thus each user, can have multiple H-ARQ processes at the same time.

Normally, the number of processes is chosen to match the round trip between the UE and the Station, including the processing time, to allow for continuous transmissions.

Typically, six processes are assigned to each user, even though it is possible up to 8. In case more processes were allowed, it would cause that the ACK/NACK response arrives with a delay not recommended.

Each user has assigned a fixed amount of memory. This memory must be distributed between the different processes. Therefore, the more processes the less memory available for each one.

Each H-ARQ transmission results in one of the following scenarios. Either it is successful decoded, or unsuccessful decoded requiring a new transmission, or unsuccessful decoded after a maximum number of retransmissions, resulting in packet error.

The H-ARQ entity should decide whether to use Chase Combining or Incremental Redundancy. An intelligent decision would be to use CC when the initial CR is low, and IR when the initial CR is high.

As it has been introduced along the thesis, H-ARQ can be configured as synchronous or asynchronous. Depending on the implementation chosen, H-ARQ retransmissions will be numbered or not, there will be a delay expected time or not, and the modulation will be able to be different from one transmission to another. All those possibilities must be covered. Details are given in CHAPTER 3.

It is important to take into account the possible errors that could happen. Here there are some basic rules. Within the downlink, an UE might be waiting for a retransmission in a concrete process. Due to a misunderstood of its last NACK sent to the BS, the BS can send a new packet, instead of the retransmission. The UE, even though it realizes there has been an error, must clear to soft buffer, and treat the new transmission as a, effectively, new transmission. However, it must inform too the higher levels, which will deal with the retransmission missed.

Another case would be when the BS misunderstands an ACK, and it sends a retransmission of the block. In that case, the UE must drop this packet, and send an ACK again, hoping this time there will be no error.

CHAPTER 6

Conclusions

This thesis evaluates the performance of a new link-to-system (L2S) interface, mainly realized through a series of mappings, which had been implemented in OpenWNS.

The main function of these mappings is to provide a block error rate (BLER) given a received coded SINR.

Before channel instantaneous conditions were taken into account in order to improve the system performance, and not OFDM/OFDMA technologies were used, a BLER was obtained directly mapping from the predicted SINR, looking at AWGN pre-stored tables.

However, due to new systems are based on exploiting channel conditions, and OFDM is used, a new physical layer abstraction is required. This new PHY abstraction must be able to model the instantaneous link layer. Moreover, a user data might be transmitted through a set of subcarriers, which might be received with different SINR. Therefore, a previous treatment of this information was needed before to be mapped into a BLER.

The new L2S interface first map the SINR value into a mutual information (MI) value. This mapping is called Mutual Information effective SINR Mapping. Once this MI value is obtained, it can be map into the final desired BLER. This map is named MI to BLER mapping along this thesis.

An evaluation of the two mappings is included in CHAPTER 1. It has been shown how the modulation coding scheme affects both mappings. A comparison against another L2S interface, used within another simulator, WiMAC, shows a considerable improvement. However, as it is explained in the précis section, the improvement is not only achieved by the use of the new mappings, but because a new coding scheme is used within OpenWNS.

After that, this thesis analyzes H-ARQ protocols. A description of these new ARQ protocols is given, as well as a few details about their implementation within different technologies.

A predicted performance evaluation has been included, regarding the two possible schemes, Chase Combining and Incremental Redundancy. It can be seen how the performance of these two schemes depends on the code rate used for the initial transmission. For higher code rates, Incremental Redundancy schemes outperform Chase Combining. However, this gain disappears progressively when the initial code rate decreases, leading to an equal performance for both schemes when the initial code rate is 1/3, due to a Turbo Codes characteristic analyzed along the thesis.

Anyways, it has been proved how the use of these new protocols improves the system performance. They do not only deal with retransmissions, but improves the throughput and might help within the link adaptation issue.

This thesis concludes that the Link-to-System interface and its mapping mechanisms implemented in OpenWNS provide the expected performance. The parameters affect the mapping as expected and shown in the literature, the general overview is positive.

A missing point that would be nice to introduce in the future is the possibility of use more than the four code rates available up to now, as well as a new coding scheme, even though Turbo Codes seems to outperform the rest.

Regarding H-ARQ, it needs to be introduced within the simulator. Some theoretical concepts and protocols characteristics given along this thesis might be helpful.

Appendix A

RBIR Calculation

This section analyzes with more detail how the mutual information value is calculated. In 2.2.1.3, it was said two approaches are considered. However, OpenWNS, thus all the simulations run along this thesis, assumes Received Bit Mutual Information Ratio. Therefore, this section will be focus on describing this approach.

A.1 Mutual Information Calculation for a SISO/MIMO System

The continuous definition of MI is:

$$MI \equiv I(X;Y) = \sum_i \int p(y \mid x_i) P(x_i) \log_2 \frac{p(y \mid x_i)}{p(y)} dy \tag{6.1}$$

According to this, mutual information per symbol can be written as:

$$MI = \log_2 N - \frac{1}{N} \sum_{i=1}^{N} E\left\{ \log_2 \left(1 + \sum_{K=1, k \neq i}^{N} \exp\left[- \frac{\left| x_i - x_k + w \right|^2 - |w|^2}{\sigma^2} \right] \right) \right\} \tag{6.2}$$

A.2 RBIR Calculation for a SISO/MIMO System and ML receivers

The symbol-level Log Likelihood Ratio (LLR) given x_i is transmitted for Maximum Likelihood (ML) receiver can be calculated as:

$$LLR_i = \log_e \left(\frac{P(y \mid x = x_i)}{\sum_{\substack{k=1 \\ k \neq i}}^{N} P(y \mid x = x_k)} \right) = \log_e \left(\frac{e^{-\frac{M_i^2}{\sigma^2}}}{\sum_{\substack{k=1 \\ k \neq i}}^{N} e^{-\frac{M_i^2}{\sigma^2}}} \right), i = 1, .., N \tag{6.3}$$

Where M_i indicates the i^{th} distances for the current received symbol, which is output from the Maximum Likelihood Decoding (MLD) detector. Therefore, $M_k = | y - Hx_k | = \sqrt{(y - Hx_k)(y - Hx_k)^H}$, where x_k represents the

k^{th} symbol, y the output of the receiver and H is the channel matrix , $H=[H_1,H_2]=\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$.

The RBIR over one constellation can be represented as:

$$I=(x,LLR)=\frac{1}{m}\sum_{i=1}^{m} I\left(x_i,LLR(x_i)\right) \tag{6.4}$$

Where $I\left(\left(x_i,LLR(x_i)\right)\right)$ is the RBIR metric of the transmitted symbol x_i over the whole constellation, and m is the number of symbols within the constellation.

Within an OFDM transmission, the MI-RBIR metric will be considered at all N sub-carriers as

$$RBIR=\frac{1}{mN}\sum_{n=1}^{N}\sum_{i=1}^{m} I_n\left(x_i,LLR(x_i)\right) \tag{6.5}$$

Let's see now how RBIR is calculated.

A.2.1 RBIR Computation-MQAM

Departing from(6.2), MI is calculated as follows:

$$\begin{aligned} MI &= \log_2 N-\frac{1}{N}\sum_{i=1}^{N} E\left\{\log_2\left(1+\exp(-LLR_i)\right)\right\} \\ &= \log_2 N-\frac{1}{N}\sum_{i=1}^{N} E\left\{\log_2 N-\log_2\left(1+\exp(-LLR_i)\right)\right\} \\ &= \frac{1}{N}\sum_{i=1}^{N} E\left\{\log_2\frac{N}{1+\exp(-LLR_i)}\right\} \end{aligned} \tag{6.6}$$

Using the expected value formula, the MI can be finally calculated as:

$$MI=\frac{1}{N}\sum_{i=1}^{N}\int p(LLR_i)\log_2\frac{N}{1+\exp(-LLR_i)}\,dLLR_i \tag{6.7}$$

In QPSK, LLR_i and LLR_k have the same pdf, due to all the symbols are equidistant between them. However, it does not happen the same for QAM in general.

It has been said that the LLR depends on the distances between symbols. Therefore, in QAM, the pdf can be approximated by calculating the LLR for the neighbouring constellation points around the one which is considered in that case. The formula applied would be:

$$LLR_i \simeq \ln\left(\frac{e^{-\frac{M^2{}_i}{\sigma^2}}}{\sum_{\substack{k\neq i \\ k\in\left\{\begin{matrix}prevalent\\ Euclidean\\ distance\end{matrix}\right\}}} e^{-\frac{M^2{}_k}{\sigma^2}}}\right) \tag{6.8}$$

Define now the RBIR as:

$$RBIR = MI / \log_2 N \tag{6.9}$$

Assuming now that the symbol level LLR satisfies the Gaussian distribution, above formula can then continuously be derived as:

$$RBIR = \frac{1}{\log_2 N}\frac{1}{N}\sum_{i=1}^{N}\int p(LLR_i)\log_2\frac{N}{1+\exp(-LLR_i)}dLLR_i \tag{6.10}$$

$$= \frac{1}{\log_2 N}\frac{1}{N}\sum_{i=1}^{N}\int_{LLR_i}\frac{1}{\sqrt{2\pi.VAR_i}}e^{-\frac{(LLR_i-AVE_i)^2}{2VAR_i}}\log_2\frac{N}{1+\exp(-LLR_i)}dLLR_i \tag{6.11}$$

It has been proved that LLR for QPSK, as well as for QAM modulation satisfies the Gaussian distribution. Therefore, the symbol LLR can be described as:

$$p\left(LLR_i\right) = N\left(AVE_i, VAR_i\right) \tag{6.12}$$

A.3 Theory derivation for Symbol LLR (LLR for QPSK symbol in a SISO scenario)

It is interesting for this thesis to look into the theory derivation of the symbol LLR, and realize how the distance between symbols determines the result.

Skipping some steps within the derivation, it is finally proved that symbol LLR can be written as:

$$LLR_i = \frac{d^2\left|h\right|^2}{\sigma^2} - K \tag{6.13}$$

Where "d" indicates the minimum inter-symbol distance in QAM constellation, h is the column vector of matrix H and k is:

$$K = \log_e\left(e^{-\frac{2d(h_r n_r + h_i n_i)}{\sigma^2}} + e^{-\frac{2d(h_r n_i + h_i n_r)}{\sigma^2}} + e^{-\frac{d^2|h|^2}{\sigma^2}}e^{-\frac{2d(h_r n_i + h_i n_r)}{\sigma^2}}e^{-\frac{2d(h_r n_r + h_i n_i)}{\sigma^2}}\right) \tag{6.14}$$

Where n_r and n_i are Gaussian, and $n_r, n_i \sim N(AVE_i, VAR_i)$

As it was mentioned above, it has been proved that LLR satisfies the Gaussian distribution.

LLR_i mean is named AVE_i and its variance VAR_i ,and they can be calculated with:

$$AVE_i = E\{LLR_i\} = \frac{d^2|h|^2}{\sigma^2} - E\{k\} \tag{6.15}$$

$$VAR_i = E\left\{\left|LLR_i - E\left(LLR_i\right)\right|^2\right\} = E\left\{k^2\right\} - E^2\{k\} \tag{6.16}$$

Where

$$E\{k\} = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\frac{d|H|}{\sigma}} \cdot e^{\frac{-x^2}{2d^2\frac{|H|^2}{\sigma^2}}} \cdot \ln\left(2e^{-x} + e^{d^2\frac{|H|^2}{\sigma^2}}e^{-2x}\right)dx \tag{6.17}$$

$$E\left\{k^2\right\} = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\frac{d|H|}{\sigma}} \cdot e^{\frac{-x^2}{2d^2\frac{|H|^2}{\sigma^2}}} \cdot \ln^2\left(2e^{-x} + e^{d^2\frac{|H|^2}{\sigma^2}}e^{-2x}\right)dx \tag{6.18}$$

From (6.14) it can be seen than *k* decreases when *d* increases, thus LLR increases with *d* (6.13). If LLR increases, RBIR increases.

In conclusion, the more distance between symbols within a constellation, the higher mutual information per bit.

Details for the LLR calculation for MIMO schemes can be found in [18].

A.4 Procedure for RBIR PHY Mapping for SISO System under ML Receiver

1. Given the channel matrix H and the SINR for each sub-carrier, the fixed LLR distribution parameter (AVE, VAR) can be computed, using (6.15) and (6.16).
2. Calculate the RBIR metric based on RBIR definition, using the LLR distribution calculated above.
3. Average the RBIR values over the multiple subcarriers for an OFDM system.
4. Convert the average RBIR for one resource block to one single effective SINR from the SINR to MI table.
5. Lookup the AWGN table to get the predicted BLER.

Again, details about the procedure for MIMO schemes can be found in [18].

A.5 Numerical example of RBIR computation

Let's calculate some MIB (RBIR) values for QPSK, given a SINR value, following the procedure presented in the section above. Note that for QPSK, $N = 4$ and $d = \sqrt{2}$.

In order to facilitate the calculation, it will be assumed that $H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Given and SINR=-3dB, we have $\sigma^2 = \frac{1}{2SINR} = \frac{1}{2 \cdot 10^{-0.3}} = 1$. We use then formulas (6.17) and (6.18), obtaining the following results: $E\{k\} = 0.8321$ and $E\{k^2\} = 3.19454$.

Then (6.15) and (6.16) are used, obtaining the following results: $AVE = 1.1679$ and $VAR = 2.5021$.

Before finally calculate the RBIR, we use (6.8) to calculate the LLR for QPSK. Note that it is the same for the 4 symbols within the constellation.

$$LLR = \ln\left(\frac{e^{\frac{0}{\sigma^2}}}{2e^{\frac{(\sqrt{2})^2}{\sigma^2}} + e^{\frac{2^2}{\sigma^2}}} \right) = 1.8730.$$

Finally, formula (6.11) is used and RBIR is calculated, obtaining $RBIR = 0.317$.

Same procedure has been used given a SINR=0dB, obtaining a $RBIR = 0.516$.

Next figure shows how these points fit the SINR to MIB curve presented along the thesis.

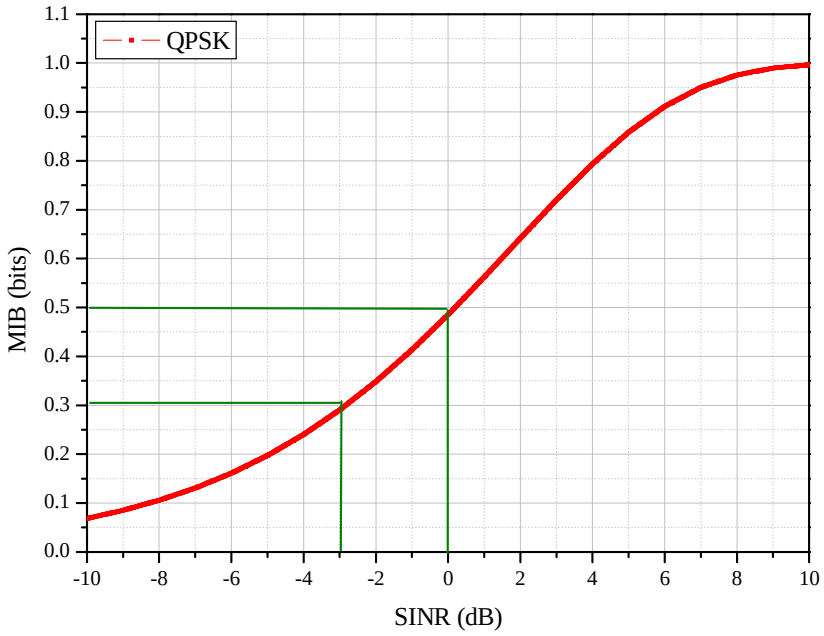

Figure 44 *SINR to MIB curve for QPSK*

Appendix B

Soft Combining

Within H-ARQ schemes that incorporate soft combining, information from failed attempts is stored and included in an iterative manner when decoding retransmissions.

There are different techniques to combine the bits, but all of them follow a similar structure. In here, one of the approaches, presented in [19], and which seems to overcome most of the rest, will be presented as an example to understand how the combination is performed.

The following notation is adopted. First, $u^{(d)} = \left[u_1^{(d)}, .., u_i^{(d)}, .., u_n^{(d)} \right]$ denotes the d^{th} transmitted copy of the codeword u in an (n,k) linear block code C over a Galois Field GF(2).

After m copies have been transmitted, including the initial one, the entire transmitted sequence can be written as $U = \left[u^{(1)}, .., u^{(d)}, .., u^{(m)} \right]$. U can be considered as a codeword in an (N,k) linear block code $C^{(m)}$ over GF(2), where $N = mn$.

Additionally, $v^{(d)} = \left[v_1^{(d)}, .., v_i^{(d)}, .., v_n^{(d)} \right]$ denotes the noisy observation of $u^{(d)}$ at the receiver. Therefore, the entire noisy version of the transmitted sequence U can be written as $V = \left[v^{(1)}, .., v^{(d)}, .., v^{(m)} \right]$. It is assumed that the transmissions are performed over a memoryless channel, and the information bits are statistically independent.

The proposed soft combining method is obtained by considering the decoding of V using the symbol maximum a priori probability (MAP) decoder for $C^{(m)}$.

The Log Likelihood Ratio (LLR) at the decoder output can be written as

$$LLR\left(\hat{u}_i^{(d)}\right) = \log \sum_{\substack{U \in C^{(m)}, \\ u_i^{(d)}=0}} P(U|V) - \log \sum_{\substack{U \in C^{(m)}, \\ u_i^{(d)}=1}} P(U|V) \qquad (7.1)$$

Where $1 \le i \le n$, $1 \le d \le m$, and $P(U|V)$ denotes the probability that U was transmitted given that V was received. $LLR\left(\hat{u}_i^{(d)}\right)$ provides an estimation of the i^{th} bit in the d^{th} transmitted copy of u . If $LLR\left(\hat{u}_i^{(d)}\right) \ge 0$ then the estimation assumes $u^{(d)}{}_i = 0$. Otherwise, $u^{(d)}{}_i = 1$.

With the assumptions mentioned above, the LLR can be written in a suitable form for performing soft combining in an H-ARQ scheme as follows. First, the LLR for the i^{th} bit of u on the m^{th} decoding attempt is defined as:

$$LLR\left(\hat{u}_i^{(m)}\right) \triangleq LLR^{(m)}\left(u_i\right) + LLR\left(v_i^{(m)} \mid u_i^{(m)}\right) + LLR_e^{(m)}\left(\hat{u}_i\right) \tag{7.2}$$

Where $LLR^{(m)}\left(u_i\right)$ is the a priori value, $LLR\left(v_i^{(m)} \mid u_i^{(m)}\right)$ represents the soft output of the channel, and $LLR_e^{(m)}\left(\hat{u}_i\right)$ is the extrinsic value, which represents extra knowledge gleaned from the decoding process, and depends on the particular structure of the code C. $LLR^{(m)}\left(u_i\right)$ and $LLR_e^{(m)}\left(\hat{u}_i\right)$ can be determined as follows. For $m=1$, the a priori values $LLR^{(1)}\left(u_i\right)$ are given by:

$$LLR^{(1)}\left(u_i\right) = \begin{cases} L\left(u_i^{(1)}\right); 1 \le i \le k \\ 0; k \le i \le n \end{cases} \tag{7.3}$$

Where $L\left(u_i^{(1)}\right)$ denotes the a priori value of the information bit $u_i^{(1)}$. For consequents decoding attempts, i.e. for $m \le 2$:

$$LLR^{(m)}\left(u_i\right) = LLR^{(m-1)}\left(u_i, v_i\right) = LLR^{(m-1)}\left(u_i\right) + LLR\left(v_i^{(m-1)} \mid u_i^{(m-1)}\right) \tag{7.4}$$

The extrinsic values $LLR_e^{(m)}\left(\hat{u}_i\right)$ are obtained for $m \ge 1$ as:

$$LLR_e^{(m)}\left(\hat{u}_i\right) = \log \frac{\sum_{r=0}^{2k-1}\left(1-u_{r,i}\right)\prod_{j=1, j\ne i}^{n} e^{\left(1-u_{r,j}\right)L\left(u_j, v_j\right)}}{\sum_{r=0}^{2k-1} u_{r,i}\prod_{j=1, j\ne i}^{n} e^{\left(1-u_{r,j}\right)L^{(m)}\left(u_j, v_j\right)}} \tag{7.5}$$

Where the joint LLR $LLR^{(m)}\left(u_j, v_j\right)$ is given by:

$$LLR^{(m)}\left(u_j, v_j\right) = LLR^{(m)}\left(u_j\right) + LLR\left(v_j^{(m)} \mid u_j^{(m)}\right) \tag{7.6}$$

If a retransmission is required, the receiver only needs to store the a priori values, in order to use them on the subsequent decoding attempt. That can be seen in (7.2). All the previous soft outputs of the channel are accumulated within these a priori values.

Appendix C

OpenWNS

OpenWNS is an open source highly modular simulator for performance evaluation of mobile radio networks [20]. It is written in C++, with Python support for configuration. It is being mainly developed at ComNets Institute, RWTH Aachen University.

The goal of OpenWNS is to provide a simulator which is suitable for various kinds of performance evaluation of wireless networks. OpenWNS a flexible framework which allows the researcher to adapt the grade of complexity/accuracy of the simulation model to his needs.

OpenWNS is made up by different parts. The main ones are:

- Main simulator executable (WNS-core)

- DLL-base: basic functionalities common to all Data Link Layers

- Radio Interference Simulation Engine (RISE), including current PHY Abstraction

- Library called libWNS, including among other generic ARQ protocols.

Some modules have been already implemented, and can be easily used in order to build the protocol stack of a station.

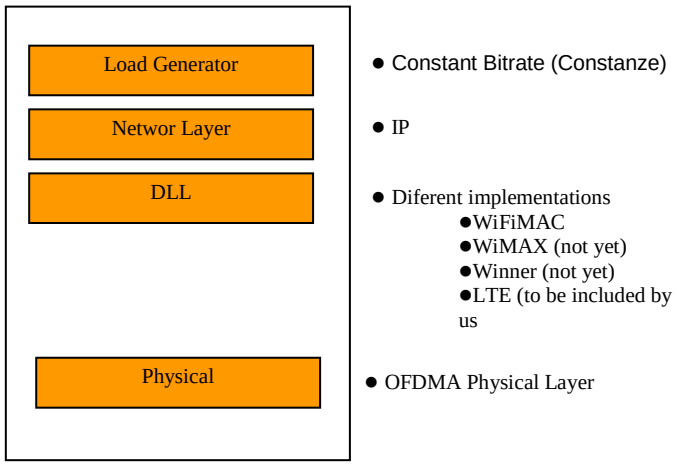

Figure 45 *Example of modules available in OpenWNS*

The configuration code in Python will allow the researcher to give the modules the exact characteristics he is looking for.

FunctionalUnits can be found, added or modified inside DLL module. FunctionalUnits are implemented functionalities that can be considered as a black box with inputs and outputs. The advantage is that they can be copied and used in several modules due to their independence of the rest of the code where they are finally located. Therefore, the DLL modules shown in Figure 45 can present the aspect exhibit in Figure 46 .

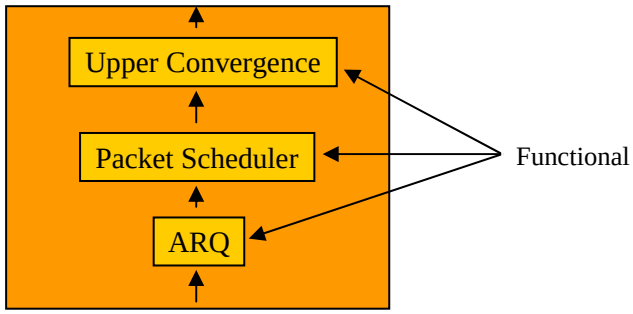

Figure 46 *Example of FU included within a DLL module*

Appendix D

Turbo Codes

This section wants to provide information about Turbo Codes and their characteristics in order to support some statements used along the thesis to explain the performance of some implementations. This section, therefore, does not want to be a guide of Turbo Codes, but a source of some details that determine the Turbo Codes performance. The section is organized as follows. First, a brief description of the two main types of Turbo Codes is given, remarking the differences between them. Finally, a peculiar characteristic of the Turbo Codes that will be used within LTE technology is explained.

D.1 Turbo Codes

D.1.1 Convolutional Turbo Codes (CTC)

The Convolutional turbo coder consists of a parallel concatenation of recursive systematic Convolutional encoders separated by a pseudo-random interleaver [2]. The stream of input bits is fed to the first encoder without any modifications and is randomly interleaved for the second encoder.

D.1.2 Block Turbo Codes (BTC)

Block Turbo code is a serial concatenation of linear codes such as Hamming, and it is decoded iteratively by simple component decoders.

D.1.3 Performance evaluation of Convolutional and Turbo Codes

The bit error rate (BER) curve of a turbo code is divided into two regions. The first region, called waterfall region, in which the BER decreases rapidly at low SINR, and a second region, called error-floor region, where the BER decreases at low rate at high SINR [15].

The waterfall region compresses BER from 1 to 10^{-5}, so it is the one interesting for this thesis, since mobile communication allow BER so much high.

The performance comparison of CTC and BTC within this region shows that CTC outperforms BTC, providing a gain of around 0.5-0.7dB, depending on the block length used [15]. This is one of the results used within this thesis, specifically in 4.3.2.1.

Further information, advantages of CTC and BTC among other concepts, are discussed in [15].

D.2 Turbo Codes in LTE

All along 5.2.2, where Incremental Redundancy scheme performance was studied, a peculiar characteristic of Turbo Codes was used. Some details have been already given, but this section will try to explain it better.

The turbo coder/encoder to be used within LTE technology is a 1/3. That means that given the payload bits that want to be sent, they first will be always coded with the Turbo coder, and a codeword with 3·payload will be obtained. If a higher code rate wants to be finally used to send the information, a puncturing of the codeword 1/3 will be performed, achieving the desired code rate. No lower code rates than 1/3 are considered in LTE. The reason is that the coding gain curve, expressed in Eb/N0, shows that the coding gain provided by lower code rates than 1/3 is almost the same as the one provided by the compulsory 1/3 code rate. That would be the first reason why the coder/decoder developed for LTE is a 1/3.

Moreover, it has been shown that the performance achieved by those higher code rates generated from puncturing the codeword 1/3 is almost the same as the performance that would be achieved in case a special coder would have been built to create a codeword with this particular code rate.

As it has been said, this characteristic of Turbo Codes has determined the evaluation scenario designed in 5.2.2.2 to test Incremental Redundancy H-ARQ schemes.

LIST OF FIGURES

Figure 1 Example of a digital communication chain [1] 3
Figure 2 Link performance model [4] 5
Figure 3 Link Layer abstraction procedure [5] 6
Figure 4 Effective SINR Mapping 7
Figure 5 BLER mapping, assuming it provides a BLER from a SINR value or MI value [5] 11
Figure 6 Example of ARQ protocols. From left to righ: Stop and Wait, Go Back N, Selective Repeat 14
Figure 7 H-ARQ scheme protocol [10] 15
Figure 8 Example Chase Combining H-ARQ [12] 16
Figure 9 Example of redundancy versions created from mother code 17
Figure 10 Example Incremental Redundancy H-ARQ process [12] 18
Figure 11 Example Codeword with CR=1/3 [13] 19
Figure 12 Example of IR scheme proposed in WINNER Project [13] 20
Figure 13 Example of a type 3 H-ARQ process 20
Figure 14 Asynchronous H-ARQ [12] 22
Figure 15 Synchronous H-ARQ [12] 23
Figure 16 Computational procedure for MIESM method [6] 27
Figure 17 SINR to MIB relation for different modulation using tables in OpenWNS 29
Figure 18 SINR to MIB relation for different modulation schemes 30
Figure 19 MIB to BLER for different MCS and BL, comparing OpenWNS and parametric function performance 31
Figure 20 MIB to BLER relation for different modulation. CR=1/2, BL=256 bits. ..33
Figure 21 SINR to BLER relation for QPSK, 16QAM and 64QAM using different CR. BL=1024 bits 34
Figure 22 SINR to BLER relation for QPSK, 16QAM and 64QAM using different BL(bits), CR=1/2. 36
Figure 23 SINR to Throughput relation for different modulations. CR=1/2 and BL= 1024. 37
Figure 24 SINR to BLER relation for different MCS. BL=256 bits. Values from WiMAC tables 39
Figure 25 SINR to BLER relation for different MCS. BL=256 bits. Values from OpenWNS tables. 39
Figure 26 SINR to BLER relation for different MCS. BL=1024 bits. Values from WiMAC tables 40
Figure 27 SINR to BLER relation for different MCS. BL=1024 bits. Values from OpenWNS tables. 41
Figure 28 PHY processing for DL-SCH [12] 42
Figure 29 SINR to BLER relation for QPSK and 64QAM, allowing up to 3 CC retranmissions. CR 1/2, BL 512 bits 46
Figure 30 Througput for 64QAM CR 5/6 , BL 512, using or not CC, allowing up to 5 transmissions 47
Figure 31 Comparison between throughput achieved by using first or second scenario. 64QAM CR 1/2 BL 1024 bits. Zoom in right figure 48
Figure 32 Example of a codeword where no bits will be sent more than once 49
Figure 33 Model of the scenario implemented 52

Figure 34 SINR to BLER relation for 64QAM, allowing up to 2 IR retranmissions. CR 5/6, BL 512 bits....53
Figure 35 SINR to BLER relation for 64QAM, allowing up to 2 IR retranmissions. CR 1/2, BL 512 bits....54
Figure 36 Example of redundancy version sent for IR retransmission with CR 1/3 ..55
Figure 37 SINR to BLER relation for 64QAM, allowing up to 2 IR retranmissions. CR 1/3, BL 512 bits....55
Figure 38 Througput for 64QAM CR 5/6 , BL 512, using or not IR, allowing up to 5 transmissions 56
Figure 39 Example of IR effectiveCR after retransmissions starting from CR 1/2 or 5/6....57
Figure 40 SINR to BLER for 64QAM. Initial CR 5/6, BL 512 58
Figure 41 SINR to BLER for 64QAM. Initial CR 1/2, BL 512 58
Figure 42 SINR to BLER for 64QAM. Initial CR 1/3, BL 512 59
Figure 43 Throughput achieved for 64QAM and 3 different initial CR. BL 512. Up to 4 tranmissions allowed 60
Figure 44 SINR to MIB curve for QPSK 68
Figure 45 Example of modules available in OpenWNS....71
Figure 46 Example of FU included within a DLL module....72

LIST OF TABLES

Table 1 β values obtained after simulation assuming channel model ITU pedestrian 1, and channel model ITU pedestrian 2. 10
Table 2 Different MCS used within the simulation in order to compare the performance of diverse ESM. 24
Table 3 σ values obtained using different ESM approaches and MCS.....25
Table 4 SINR to MIB values for different modulations included in [6] (left). Same values from OpenWNS tables (right) 28
Table 5 SINR values to achieve a BLER = 0.1 for different modulations and different CR. BL=1024 bits. Inc denotes the increment in terms of SINR when using different CR. 35
Table 6 SINR values to achieve a BLER = 0.1 for different modulations and different BL. CR=1/2. $|Inc|$ denotes the absolute value of the increment in terms of SINR when using different BL. 36
Table 7 SINR values when BLER=0.1 for different MCS and BL=256, using values from WiMAC and OpenWNS. Comparative. 40
Table 8 SINR values when BLER=0.4 for different MCS and BL=1024 bits , using values from WiMAC and OpenWNS. Comparative 41
Table 9 Standard deviation in terms of throughput for different percentages assuming second scenario.....48

GLOSSARY

ACK	Acknowledge
ARQ	Automatic Repeat Request
BL	Block Length
BLER	Block Error Rate
BPSK	Binary Phase-Shift Keying
BS	Base Station
CC	Chase Combining
CR	Code Rate
Eb	Energy per Bit
ESM	Effective SINR Mapping
EESM	Exponential Effective SINR Mapping
FEC	Forward Error Correction
H-ARQ	Hybrid Automatic Repeat Request
HSPA	High Speed Packet Access
HSDPA	Downlink HSPA
HSUPA	Uplink HSPA
IR	Incremental Redundancy
LLR	Log Likelihood Ratio
LTE	Long Term Evolution
L2S	Link-To-System
MAC	Media Access Control
MCS	Modulation Coding Scheme
MIB	Mutual Information per Bit
MIESM	Mutual Information Effective SINR Mapping
MIMO	Multiple Inputs Multiple Outputs
MMIB	Mean Mutual Information per Bit
NACK	Negative Acknowledge
OFDM	Orthogonal Frequency Division Multiplexing
OpenWNS	Open Wireless Network Simulator
PER	Packet Error Rate
PHY	Physical Layer
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
RLC	Radio Link Controller
RNC	Radio Network Controller
RRBIR	Received Bit Mutual Information Ratio
SI	Symbol Information
SINR	Signal to Interference plus Noise Ratio
UE	User Equipment
UMTS	Universal Mobile Telecommunication System
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network

BIBLIOGRAPHY

[1] Karsten Brueninghaus, David Astely and Thomas Sälzer. *Link Performance Models for System Level Simulations of BroadBand Radio Access Systems*. IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communication. P 2306-2311 Vol. 4. Berlin. 2005.

[2] Claude Berrou, Alain Glavieux and Punya Thitimajshima. *Near Shannon Limit Error – Correcting Coding and Decoding: Turbo Codes (1).* Proceedings of IEEE International Communications Conference. 2003.

[3] Jordi Gomez Garcia. *Performance Evaluation of Fast Link Adaptation Algorithms in MIMO Wireless LANs*. Master Thesis. Aachen, Germany. 2008.

[4] Martti Moisio and Alexandra Oborina. *Comparison of Effective SINR Mapping with Traditional AVI Approach for Modelling Packet Error Rate in Multi-state Channel*. Y.Koucheryavy (ed). Springer-Verlag Berlin Heidelberg. P 461-473. 2006.

[5] Xin He, Kai Niu, Zhiqiang He and Jiaru Lin. *Link layer abstraction in MIMO-OFDM system*. IWCLD International Workshop on Cross Layer Design p 41-44. 2007.

[6] *IEEE 802.16m Evaluation Methodology Document (EMD).* Project IEEE 802.16 BroadBand Wireless Access Working Group. IEEE 802.16m-08/004r3. 2008

[7] Erik Westman. *Calibration and Evaluation of the Exponential Effective SINR Mapping (EESM) in 802.16*. Master Thesis. Stockholm, Sweden 2006.

[8] John G. Proakis. *Digital Communications*. Osborne-McGraw Hill. Third edition. New York. 1995

[9] Kian Chung Beh, Angela Doufexi and Simon Armour. *Performance Evaluation of Hybrid ARQ Schemes of 3GPP LTE OFDMA System.* Personal, Indoor and Mobile Radio Communications. PIMRC 2007. IEEE 18th International Symposium. P 1 – 5. 2007

[10] Ingo Viering. *System Aspects in Communications.* TU München lecture. 2008

[11] Ki-Ho Lee, Gyung-Ho Hwang and Dong-Ho Cho. *Type II Hybrid ARQ Scheme based on QoS and LDPC Code*. Vehicular Technology Conference. VTC2004-Fall. IEEE 60th. P 2630 - 2634 Vol. 4. 2004.

[12] Erik Dahlman, Stefan Parkvall, Johan Slöld and Per Beming. *3G Evolution. HSPA and LTE for mobile broadBand*. Elsevier. Oxford, UK. First Edition 2007.

[13] Thierry Lestable, Yi Ma, Monica Navarro, Adam Piątyszek, Stephan Pfletschinger (editor), Christian Senger, Mikael Sternad, Tommy Svensson. *D2.2.3 Modulation and Coding schemes for the WINNER II System*. Winner Project. 2007.

[14] Pal Frenger, Stefan Parkvall and Erik Dahlman. *Performance Comparison of H-ARQ with Chase Combining and Incremental Redundancy for HSPDA*. IEEE Vehicular Technology Conference. VTC 2001 Fall. Vol. 3. P 1829-1833. USA 2001.

[15] K. Ramasamy et.all. *Performance Comparison of Convolutional and turbo Codes*. IEICE Electronics Express. Vol. 3. No 13. P 322-327. 2006

[16] Jung-Fu Cheng. *Coding Performance of Hybrid ARQ Schemes*. Communications, IEEE Transactions on P 1017-1029. 2006.

[17] Christian Hoymann. *IEEE 802.16 Metropolitan Area Network with SDMA Enhancement*. Doctorate Thesis. 2008.

[18] Hongming Zheng et. all . *Link Performance Abstraction for ML Receivers based on RBIR Metrics*. IEEE 802.16 BroadBand Wireless Access Working Group. 2007.

[19] I.D Holland, H.J Zepernick and M. Caldera. *Soft Combining for Hybrid ARQ. Electronics Letters*. Vol 41 No 22. 2005

[20] OpenWNS project Homepage. http://www.openwns.org/Wiki