Czech Technical University in Prague
Faculty of Electrical Engineering

MASTER'S THESIS

2014
Petr Procházka

Czech Technical University in Prague
Faculty of electrical engineering
Department of Radio Engineering

Experimental Wireless Cloud TxR Testbed Platform

May 2014

Author: Petr Procházka
Supervisor: prof. Ing. Jan Sýkora, CSc.

Proclamation

I declare that I have worked on my diploma thesis independently and only sources stated in bibliography were used. I agree that my thesis or its parts can be borrowed and provided to public.

Date: 12.5 2014

.....

Signature

Task

Acknowledgement

I would like to thank to prof. Ing. Jan Sýkora for his very wise and useful advices and comments, which helps to create this thesis. Also I would like to thank to all doctoral students from DiRaC group for their help with many obstacles and for very pleasant and friendly atmosphere establishment.

Abstract:

At this time Wireless Physical Layer Network Coding (WPNC) is a hot topic in research community as a means to enhance the overall network throughput. The main goal of the thesis is to develop, practically build and verify the experimental wireless cloud network platform with multiple Tx/Rx for software radios. The platform should be configurable to serve various topology and processing scenarios including possibility to emulate simplified processing options such as perfect synchronization or symbol timing. This will involve the development of AirInterface wrapper protocol in order to define basic rules for communication in wireless network. The platform should have a user interface which allows to create a signal from data inputs and also direct work with signal samples (IQ BB) for offline processing. Platform should be designed in such a way to provide some smaller tasks as various modulations, NCM encoder, H-decoding, synchronization options etc. The viability of the platform should be demonstrated by example implementations and real radio hop experiments of selected (both point-2-point, and cloud oriented) scenarios with simple forms of WPNC components.

Key words:

Wireless Physical Layer Network Coding, Synchronization, Platform, AirInterface protocol

Abstrakt:

V současné době je Wireless Physical Layer Network Coding (WPNC) středem zájmu výzkumníků v oblasti bezdrátových komunikací jako jedna z možností jak docílit výrazného zvýšení propustnosti těchto sítí. Cílem této práce je navrhnout, vytvořit a posléze ověřit experimentální platformu pro bezdrátové sítě tvořenou softwarovými radii, která umožní spolupráci více přijímačů a vysílačů. Tato platforma by měla sloužit k nastavení různých topologií sítě včetně možnosti emulace ideálních procesů typu dokonalá synchronizace nebo symbol timing. Dále by měl být vytvořený protokol pravidel pro komunikaci v bezdrátových sítí. Platforma by dále měla obsahovat uživatelské rozhraní umožnující vytvoření požadovaného signálu ze vstupních dat a také práci přímo se vzorky již přijatých signálů pro offline zpracování. Vytvořená platforma by měla umožňovat další možnosti v podobě různých forem modulací, synchronizačních možností nebo hierarchického kódování či dekódování. Součástí této práce je také ověření funkčnosti této platformy na názorných ukázkách realizace různých scénářů topologie sítě včetně základních prvků WPNC.

Klíčová slova:

Wireless Physical Layer Network Coding, Synchronizace, Platforma, Modulace,

Contents

Nomenclature

3GPP	3rd Generation Partnership Project
AP	Access Point
AWGN	Additive White Gaussian Noise
CAZAC	Constant Amplitude Zero Auto-Correlation
CFO	Carrier Frequency Offset
CSE	Channel State Estimation
DF	Decode and Forward
DSP	Digital Signal Processor
HNC	Hierarchical Network Code
LTE	Long-term evolution
MC	Mutually coupled
MS	Master-slave
NC	Network Coding
OFDM	Orthogonal Frequency-Division Multiplexing
PHY	Physical Layer
PiAcq	Acquisition Pilot
PiCSE	CSE Pilot
PiHrc	Hierarchy Pilot
PL	Payload
SNR	Signal-to-Noise Ratio
WNC	Wireless Network Coding
WPNC	Wireless Physical Layer Network Coding

List of Figures

Part I

Theoretical background

Kapitola 1

Introduction

In today's modern society, wireless communications is one of the most active areas of technology development, and the fastest growing communication - based division of our time. Wireless communication has been developed to provide societiy services for different applications and purposes. This developement made a huge progress from it's very beginning where wireless was just a medium for supporting vioce telephony up to a medium for supporting other services, for instance transmission of data streams or internet connectivity. Thus, similarly to development in wired line capacity in the 1990s, the demand for current wireless capacity is rapidly rising. But these increasing demands rise many others technical problems to be solved. In case of wired line capacity, where the demand can be fulfilled for instance with addition of new wired line, or new infrastructure such as optic fibre and so on. But in case of wireless systems the only resources that have been used to increase capacity are transmitter power and radio bandwidth. Unfortunately, these two resources are in fact very restricting.

Transmitter power can be higher, but when we consider the fact that most of the wireless devices (phones, laptops or tablets) require the use of battery supply, it make this classical trade-off problem between performance and device live time. So this solution would not be optimal. Second option - Radio bandwidth is also very resticting, with regard to useful radio bandwidth, which is limited by number of devices (common bandwidth) or it's very expensive to rent extra bandwidth. Main reason of increasing demand on capacity of wireless networks is extreme growth of mobile devices (smartphones, laptops, tablets etc.) over the world.

The increasing number of wireless devices that are accessing wireless networks worldwide is one of the primary contributors to global wireless data traffic growth. Each year several new devices in different form factors and increased capabilities and intelligence are being introduced in the market, for instance GPS systems in cars, asset tracking systems in manufacturing sectors and shipping, or medical applications to provide patient records and status more available. This forecast also predicts that North America and Western Europe are going to have the fastest growth in mobile devices and connections. If few years ago

was one device using wireless networks mean per person, in very close future it could be doubled or maybe tripled due to relatively low cost of mobile devices and market expansion. I allowed to gather two exemplary figures to prove my arguments from Cisco Global Mobile Data Traffic Forecast [14]. First figure 1.1 shows growth in number of devices accessing wireless networks which directly affects wireless data traffic growth depicted in figure 1.2. This growth of wireless data trafic with time seems to be exponential for the upcoming few years. These circumstances motivate engineers all over the world to develope methods how to improve wireless network throughput and channel capacity to satisfy very rapidly growing demand on wireless data transfers.

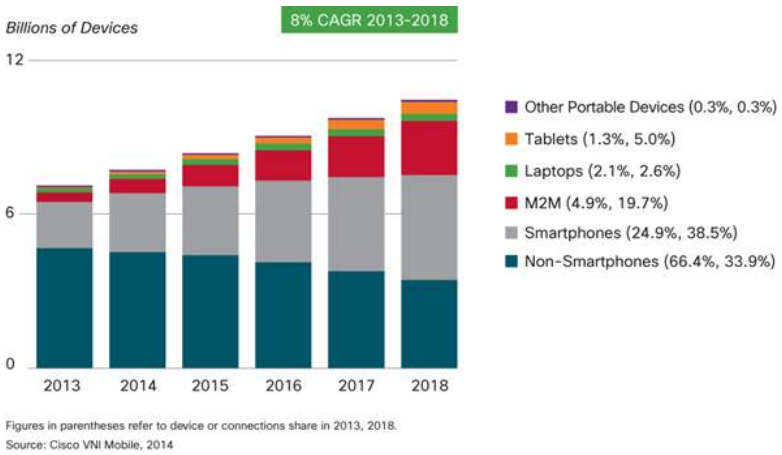

Figure 1.1: Global Mobile Devices and Connections Growth [14]

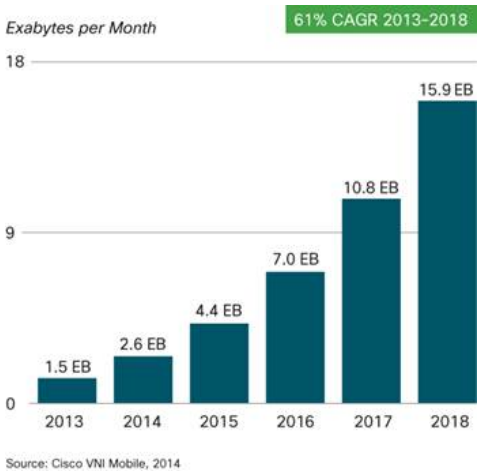

Figure 1.2: Wireless data traffic growth forecast [14]

Nevertheless there is one positive fact left, processing power. Moore's Law, which predicts doubling of processor capabilities every two years, has been quite accurate over the past 20 years and its accuracy promises to continue for upcoming years. Given by these circumstances, there has been considerable research effort in recent years focused to enhance the wireless capacity and throughput to satisfy increasing demands without attendant increases radio bandwidth or power requirements.

1.1 Motivation

Traditional approach of the routing information is that routers are operating with „store and forward“ strategy, which is very simple.The node (router) is only deciding whether the recieved information is intended to it then send it further to the next hop device, and if it's not it decide where to sent it further on. The information is routed by the network nodes from the sources through a web of the conncected networks till it reaches it's destination. They just buffer received packets and forward them (almost)unmodified, but the information carried by the packet is the same all the way from source to destination.

Scenario of common wireless network can be decomposed to a set of distinguishable point-to-point oriented nodes. Destination point is prepared only for signal from one particular device and all other signals are treated as harmful interferences and have to be prevented by the processing procedures. But these suppressed signals can obtain very useful information, for instance information about network topology, channel estimation etc.

When we consider these circumstances we have to ask question, if the current routing solution is the optimal one for multi nodes complex networks or if we could do it better ?

Can we built more sophisticated solutions in wireless networks for the crucial points of routing-based approach? These crucial points are:

1. Complex network nodes works only on Store & Forward strategy.
2. The whole network, no matter how complex or huge the network is, is formed by a set of point-2-point links.
3. The rest of the network is ignored (relative to picked up random point-2-point segment of complex network) included any knowledge of the topology of network we could possibly have.

All those assumptions generally degrades performance of the network and networks designed with properties like these are highly non-optimal, compared to the network performance based on better than Store & Forward strategy. Other possible strategies are shown in chapter two as we show further in this work.

In this work we try to change this standard approach and create nodes (relays) with ability to „code“ information from several sources before forwarding

them and broadcast this coded information further into wireless cloud. User at the destination point then can be able to decode original information from the sources. It can be done at different layers that means on level of packets, symbols or signals. This method improving network capacity is called Network Coding.

1.2 Thesis Outline

Second chapter of this thesis is dedicated to fundamentals about network coding and its benefits. After that is network coding principle aplicated into wireless networks. Aslo there are mentioned basic relaying strategies and introduction into Hierarchical & Froward strategy. Third chapter is focused on establishment of interface protocol to provide a communication rules in wireless networks. In the same chapter are also provided information about pilot signals used in testbed platform. Next part of third chapter is focused on implementation of framework platform into choosen environment which serves for testbed design. Main goal of fourth chapter is to create a stand-alone application from designed framework with user friendly Graphical User Interface. Chapter five is dedicated to remote access to designed framework application. Also there is mentioned another techniques of remote access. Chapter six serves to demonstration of results of various network scenarios. These results are physically recieved signals from designed testbed compared with signal model at the transmitter side in order to demonstrate negative environmental influences on signal propagation.

Chapter 2

Network Coding

This chapter is dedicated to a basic network coding idea. In introduction was sign about increasing demands on capacity of wired networks in 1990s, which among others, leads to research of new approaches of forwarding data through the networks to the destination point effectively. A significant breakthrough came in 2000 when Ahlswede introduced Network Coding (NC) [1]. Instead of conventional way of Store & Forward information in networks, which degrades its performance, Ahlswede came up with idea to mix messages from different sources into one in order to achieve multicast capacity. Section 2.1 is an introduction to network coding basic principle. In Section 2.2 we apply this new approach to Wireless networks. The next section 2.3 is dedicated to possible relaying scenarios. Section 2.4 is dedicated to Hierarchical Decode & Forward strategy. This relying strategy is significant for the rest of this work.

2.1 Network Coding principle

The best way how to demonstrate principle of Network Coding is through the famous butterfly example as shown in figure 2.1.

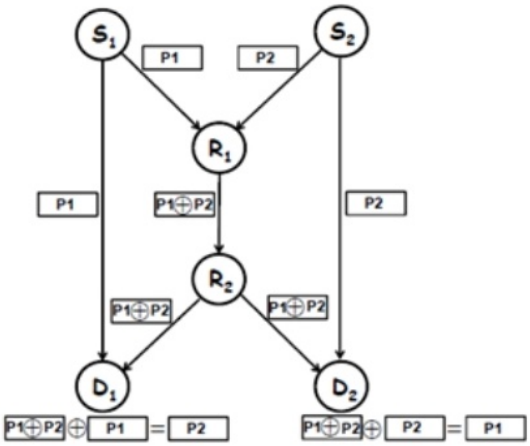

Figure 2.1: Network coding butterfly scenario example.

Scenarion in figure 2.1 can be described this way. Source S_1 wants to deliver packet P_1 to both destinations D_1 and D_2 as well as source S_2 wants to send packet P_2 to the same pair of receivers. For this example let's assume that all links have a capacity of one packet per second. If routers R_1 and R_2 only store and forward the packets they recieve, the middle link be overwhelmed cause both routers for every second can either deliver P_1 to D_2 or P_2 to D_1 but not both variants. Compare this with situation where the router feeding the middle link with XORs (XOR operation is denoted by $\oplus$) of the two packets from the sources and sends $P_1 \oplus P_2$, as its shown in figure 2.1. In that case both receivers can obtain both packets. Destination D_1 can get P_2 by XORing packet P_1 received on the direct link from S_1 with broadcasted $P_1 \oplus P_2$ from the router R_2. And analogically D_2 recovers packet from source S_1.

This solution is optimal in the sense of the maximal throughput, in this case is final throughput two packets per channel usage. This approach is simply better than the routing approach which can, at best possible scenario, achieve 1.5 packet throughput per channel usage. In this example is used XOR for the encoding routine to combine two packets from sources S_1 and S_2, but it could be replaced by a liner combination (in general it can be nonlinear combination as well, but in this work we will mention only linear combination for simplicity) of the data, interpreted as numbers over some finite field. This approach is called Linear Network Coding and provides very useful tool since the linear NC operations for encoding and decoding can be easily rewriten in form of the matrix equations. From these equation can be easily verified invertibility of global encoding function into decoding function (the inverse matrix) via the matrix ranks and tools of linear algebra [2].

Although linear encoding functions seems to be optimal for solving single

source multi-cast problems, paper [3] showed that linear NC is insufficient for solving network of many sources in general.

2.1.1 Network Coding Gain

We shown in example from section 2.1 that NC allows to internal networks increase throughput by combining incoming streams rather than the simple storing and forwarding operations. So far we were considered NC only from a theoretical point of view, but NC was also proven in practical implementation. In wired form it makes revolution in telecommunications complex networks. Also WNC was already implemented. The paper [4] defines a protocol named COPE, but it is not pure implementation of WNC because COPE is operating between Medium Access Control Layer (MAC) and network layer instead of directly at the physical layer (PHY). And this is a nowadays trend in WNC research, and also our goal in this work, to move NC principle as close as posible to PHY i.e. at level of electromagnetic waves.

2.2 Wireless Network Coding

Principle of network coding can be applied to a wide variety of scenarios not only on wired networks but also for wireless networks even for an ad-hoc mobile wireless networks. The first idea of implementing network coding on wireless network is probably in paper by Shengli Zhang et al. [5] in 2006. It's whole six years after the origin of the network coding approach [1]. The main goal of this section is to describe differences between Wireness Network Coding (WNC) and Network Coding (NC).

The very important thing is to define where (on which layer) combination of input data streams occur. In the case of wired NC is very simple to distinguish data input from several sources, because there exist "physical" wired connection with every node in network. But in case of wireless interface we are not able to distinguish incomming data inputs so simply. The main and crucial difference between WNC an NC is that WNC devices sharing same medium to communication so in case of multiple sources are transmitting simultaneously, at the PHY of communication link (i.e. receiving antenna) we are receiving superpossition of these signals. This is the most challenging issue compared to a wired communications, if we were able to handle this superpossition of incoming signals right, we can get additional potential throughput benefit.

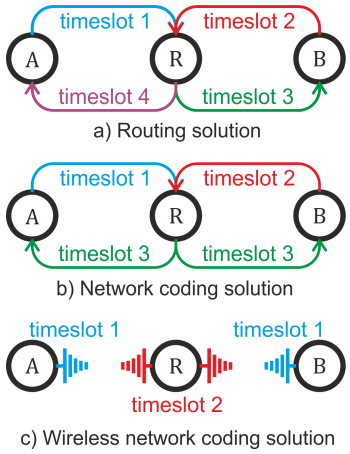

Figure 2.2: Two-way relay channel

Figure 2.2 shows all the scenarios previously mentioned now on a simply two-way relay channel. This network consist of two transmitters A, B that want to exchange information to each other via relay$R$ in such a way, that each serves as a receiver for the opposite side partner. For highlighting benefits of the WNC let's assume that different transmissions are separated into time slots for simplicity. Classical routed solution needs for delivering 2 packets 4 time slots so the throughput of this solution is $1/2$ packets per channel use. In the NC approach throughput is $2/3$ packets per channel use. In case of WNC throughput is 1 packet per channel use.

Other significant difference between NC and WNC is that WNC's domain are electromagnetic waves instead of discrete values (bits) where data $d \in \{0, 1\}$ on PHY of NC. The very last and also very important difference is in unpredictable behaviour of channel. In NC are channels (copper wires/optic fibres) mostly quite stable, error-less and interference free. On the other hand wireless channel is highly unstable and full of different types of interferences caused by multipath spreading, dispersion in frequency, phase rotation, etc.

2.3 Relaying strategies

We can divide several kinds of signal processing methods performed by the relay into three families. This procedures are called Relaying Strategies and we can separate them by the fact whether is decision to be made by the relay about incoming signal or not.

1. Amplify & Forward (AF) is a strategy where is no decision needed. Relay only scales/amplify incoming (superpossed) signal and transmits it towards the other nodes. This method is very simple, but due to amplyfying received signal we also increase its noise that is forwarded further into network.

2. Compress & Forward (CF) is a strategy where relay does not make a full decision about the signal and only partial (compressed) information is broadcasted towards the other nodes in network.

3. Decode & Forward (DF) in these strategies are some decisions needed. We can further divede this category into:

- Hierarchical Decode & Forward (HDF) strategy processes superpossed codewords - Hierarchical codewords directly in signal space representation. HDF is described in more details in next subsection.
- Joint Decode & Forward (JDF) is similar to Physical Layer Network Coding (PLNC) [6,7]. In these scenarios relay decodes all signals separately and applies a network code upon those estimates.
- Compute & Forward (CmpF) [8,9] which uses properties of lattice codes in order to process superpossed signals at PHY
- De-Noise & Forward (DNF) strategy similar to HDF but more focused on symbol by symbol relay processing with channel parametrization.

2.4 Hierarchical Decode & Forward

Purpose of this section is to shortly introduce Hierarchical Decode & Forward strategy, its basic principle and properties. This relaying strategy was firstly presented in [10], but more informations with simulation results and constrains respectively conditions of HDF are contained in [11,12,13]. In the research community, multi-node and multi-source wireless communication scenarios are under intensive investigation mainly how to effectively provide data flows and network coding by relay nodes without routing. And that's a reason why was HDF created.

Basic principle of HDF can be metaphorically and in very simplified form explain this way. Let's assume that we need create a function for nodes (relays) to provide data flows without an explicit routing. This routine can be described as a flood of the information having different "colors" (each source has data stream with unique color) at the input of the node in network. The node based on HDF relaying strategy processes the "rainbow" mixture-color from not distinguishing individual sources of data streams. The destinations pick a particular "color" from the received flood with the help of various forms of the side-information on other data streams available at the destination. Since the mixture-data flow represents jointly (but not necessarily, in more complicated networks, individually distinguished) data stream, we call those data hierarchical data.

This approach holds the principle of Network Coding, but with two major differences.

1. The information transfer is achieved through signal-space wireless links. It means that are phenomena like signal superpositions, fading and channel parametrization are included.

2. The distributed source coding joint with distributed channel coding can play important role when we don't fully decode and re-code the hierarchical codewords at relays due to the latency constraints.

We can demonstrate above mentioned concept on the simplest possible network scenario with two sources and one relay. This scenario with ilustrated Hierarchical side information and Complementary side information is shown in figure 2.3.

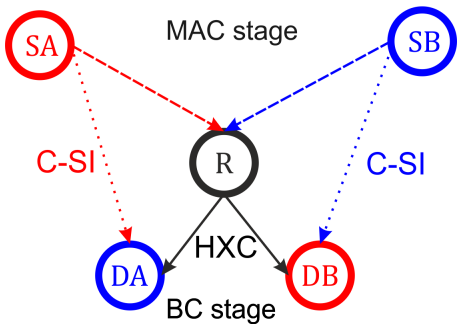

Figure 2.3: 2-Source Relay Network with side information

A 2-Source Relay Network (2-SRN) consist of two data sources SA and SB and both want to send their data to two destinations DA and DB via shared relay R. During the first (MAC) stage sources A and B simultaneously transmitting towards the relay and also destination B collects $C-SI$ on the data A and destination A $C-SI$ on the data B. The crucial feature that distinguishes HDF strategy from other Decode & Forward methods is the fact that the relay does not decide about the individual source codewords (packets, data, symbols, etc.) at all.

When we assume recieved signal (superposed signals from sources A and B) at relay input in form

$$u = h_A s_A + h_B s_B + w \tag{2.1}$$

where h_A, h_B are appropriate channel scaling coeficients and w is circularly symmteric complex Additive White Gaussian Noise (AWGN) with variance σ_w^2 per complex dimension. The only decision that is made by the relay is directly on signal space representation and this decision is about so called hierarchical symbols. The rule for relay's decision making is strictly given by

$$\hat{d_{AB}} = \arg\max_{d_{AB}} \mu(d_{AB}) = \arg\max_{d_{AB}} \mu\left(\bigcup_{d_A, d_B : \chi(d_A, d_B) = d_{AB}} \{d_A, d_B\} \right) \tag{2.2}$$

Maximum of metric $\mu(.)$ is sought among hierarchical symbols d_{AB} not like at others Decode & Forward methods (namely JDF) where is metric $\mu(.)$ sought among pair of individual source symbols d_A, d_B. This is very important to noticed, for comparison with JDF is shown decision rule of JDF relaying strategy

in eq.2.3. Individual symbols d_A and d_B are "combined" into the hierarchical symbols d_{AB} by so-called Hierarchical Network Code (HNC) fuction $\chi(.)$.

$$\left[\hat{d_A}, \hat{d_B}\right] = \left[\arg \max_{d_A, d_B} \mu\left(d_A, d_B\right)\right] \tag{2.3}$$

To achieve ability to decode the imposed HNC at the destination the applied HNC have to fulfil a condition called an exclusive law [10,11]. This exclusive law is defined as a property of the joint representation of two data symbols through the function $\chi_d(d_A, d_B)$. It must hold that

$$\chi_d(d_A, d_B) \neq \chi_d(d'_A, d_B),\ \forall d_A \neq d'_A, \tag{2.4}$$

$$\chi_d(d_A, d_B) \neq \chi_d(d_A, d'_B),\ \forall d_B \neq d'_B, \tag{2.5}$$

The functions fulfilling exclusive law will be called exclusive maapping and denoted by the operator $\chi_d(.\,,.)$ specifying also the corresponding symbol for which it applies (in this case d). In other words the exclusive mapping allows invert the mapping function provided with a side information on one of the data symbols. Assuming that the destination node B has perfect H-SI on the node's own data d_B it can then decode the message d_A(and analogicaly for node A). We will call data d_B complementary data from the perspective of the data d_A operations. The exclusive mapping can be applied at various levels for instance in a level of symbols we define Hierarchical eXclusive Alphabet (HXA). But in this example on a level of data symbols it is defined like a Hierarchical eXclusive Code (HXC) as two-source codebook fulfilling the exclusive law.

The exclusive law can be extended to the complex scenarios of much more complicated networks that (2-SRN), but the number of equations grows immensely. In this work we will not discuss about methods how to select appropriate HNC at each node in the network neither how to check invertibility of the HNC. These topics are shown in [11,12,13] and related works.

It seems that HDF offers significant advantage, because HDF signal processing is much more complex. In [11] is shown that HDF can offer a rectangular capacity region that can exceed beyond the capacity regions of the other strategies. This example of capacity regions is shown in figure 2.4 and the capacity is shown in figure 2.5. Nevertheless this improvements are done at the expense of significantly increasing processing complexity. In addition this processing complexity grows exponentially with increasing number of devices in network.

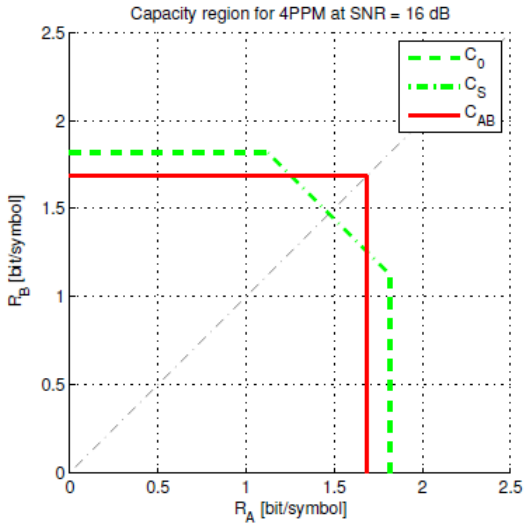

Figure 2.4: Capacity regions of HDF [13]

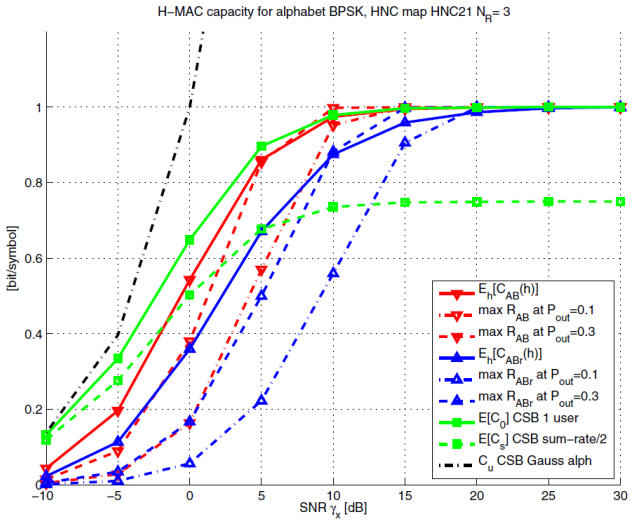

Figure 2.5: MAC capacity for HDF [12]

Part II

Practical contribution

Chapter 3

Interface protocol and framework platform

This chapter is focused on defining of interface protocol wrapper. It should be designed generic enough to serve for different network cloud scenarios used in platform. Creation of platform itself is desribed in chapter 4. The second goal of this section is choice of enviroment for this platform which provide sufficient computing performance to handle all wireless network requirements and also adequate conditions for programming part of this platform.

In beginning of this section are fundamentals about wireless network synchronization and distribution of synchronization in these networks. There are described related problems and current state-of-art approaches. Next part is dedicated to definition of interface protocol that properly addresses multiple cloud stages and provide synchronization preferences. After that is explained choice of basic enviroment for platform including possible other solutions for given network scheme. In this chapter will be also provided implementation of each approaches of transmission routines with further informations and results.

3.1 Current state-of-art network synchronization in wireless networks

Synchronization is very important in every network. In general is main target of network synchronization equal distribution of time and frequency to all devices in network. Synchronization also define symbol timing and distinguish data from a additive signals used for signalisation. It is necessary for proper communication especially from receiver point of view.

In wireless networks time and frequency synchronization also serves for a schedule of multiple access to a shared wireless medium. Generally is every device in network equiped with its own local oscilator that generates periodic signals. But these oscillators have limited stability and accuracy which cause

that every oscillator have a slightly different properties. It is commonly given by manufacture processes and temperature conditions. Therefore is necessary to coordinate all these oscillators for establishment of common frequency reference.

In wireless network are two basic approaches how to establish and distribute frequency reference.

1. Master-slave (MS) solution is based on hierarchical principle when every network or subnetwork has a node with precise reference clock (master node). The master node stays at the top of hierarchical topology and all other nodes belonging to a lower priority just adjust their local precise clocks.

2. Mutually coupled (MC) network synchronization is based on peer-to-peer arranged nodes, where each node controls its local clock with the information received from all its neighbours.

To introduce basic definitions about synchronization regimes, let's consider a simple harmonic oscillator with output

$$x(t) = \mathrm{A}(t)\sin(\omega t + \phi) \tag{3.1}$$

where ϕ is the oscillator phase at $t = 0$ in radians, f is frequency of the oscillator and t is absolute time. In this case we can neglect amplitude scaling $\mathrm{A}(t) = \mathrm{A}$ because it has no influence on frequency or phase. Then we can claim that phase of generated signal increments with a speed proportional to the oscillator frequency in rad/s ω, $d\Theta(t)/dt = 2\pi f$. Local time function associated with $x(t)$ is then

$$\tau(t) = \frac{f}{f_0}t + \frac{\phi}{2\pi f_0} \tag{3.2}$$

where f_0 is the nominal frequency of the oscillator. Actual frequency is always different from the nominal one $f \neq f_0$. It is given by heating, manufacturing processes or environmental influeces. Due to these circumstances following three possible synchronization scenarios depicted in 3.1 can occur for couple of local clocks $\tau_1(t)$, $\tau_2(t)$.

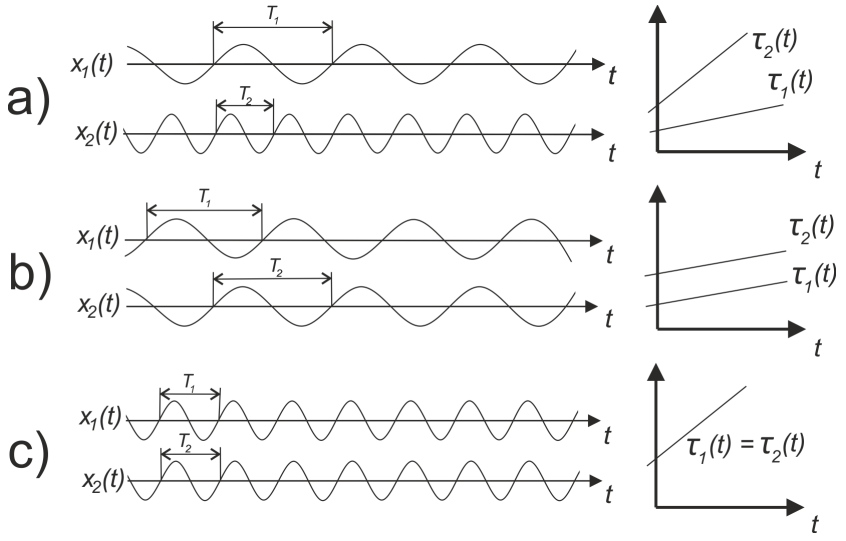

Figure 3.1: Different synchronization scenarios [15]

1. Asynchronous mode: clocks run indenpendently with different phases $\beta_1 \neq \beta_2$ and frequencies $f_1 \neq f_2$.
2. Frequency synchronization: clocks run at the same frequency f_1 but with different phases $\beta_1 \neq \beta_2$
3. Frequency and phase synchronization: phase synchronized clocks naturally implies frequency synchronization.

Most wireless communication systems in operation is based on centralised network structure, whereby access point (AP) manages the nodes that are within its transmission range. Therefore AP serves as a master node and provide natural reference for synchronization processes. In cellular based systems is necessary frequency synchronization bacause of handovers. After that could be done carrier frequency offset estimate (CFO) at each nodes from APs preamble. The time synchronization is necessary whenever medium access is organized into time slots - TDMA based communications.

The limited accuracy and stability of real oscillators have a very bad impact on transmitted signal. It's caused by difference between nominal and actual frequency of transmitter which leads to relevant signal distortion. Therefore is one of the main tasks of synchronization in nowadays wireless networks compensate carrier frequency.

3.2 Interface protocol

In digital communications generally protocol is a system of rules for data exchange between devices in network. Main purpose of protocol in communication

systems is to use well-defined formats for exchanging data messages. Each message has an exact meaning intended to provoke a particular response of the receiver. Thus, a protocol must define the syntax, semantics, and synchronization of communication. Therefore is necessary that all devices in network have to strictly keep this defined protocol as common rules for communication.

The main purpose of defining our interface protocol is to create a container of PHY solutions specific to the Wireless Network Coding based communication introduced in 2.2. It should be disigned generic enough to serve for all possible solutions and scenarios using modular implementation with respect to these key features:

1. Slot synchronization for packet based operations.
2. Frame synchronization that properly addresses multiple cloud stages.
3. Hierarchy signalisation carrying information about network scenario and other relays operations.
4. Node-local Channel State Estimation (CSE).
5. Actual useful information carried by payload (PL)

Protocol fulfils these conditions is a theoretical one, it serves as a model for implemented protocol described in next text of this document.

3.2.1 Interface protocol model

Model of ideal protocol for general wireless networks is shown in figure 3.2. It is general enough to provide all functionality depicted in section 3.2.

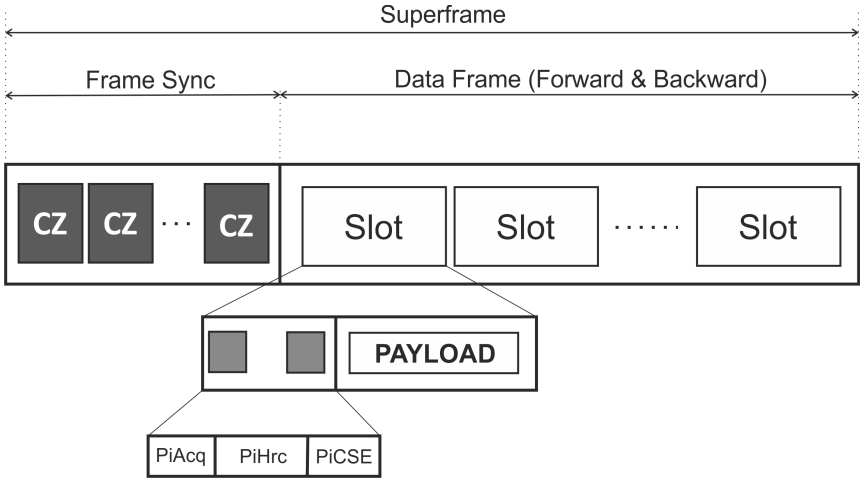

Figure 3.2: Interface protocol model [15]

Structure of ideal protocol can be diveded into two main parts. First part is signalisation of superframe start consists of number of unmodulated symbols. This part is called frame synchronization, and for this task are recommended by [15] CAZAC sequences due to its properties. The second part consists of pilot signals and modulated data symbols. From this point of view synchronization section and payload can be assumed as two separated entities.

Very important part of this model are pilot signals. These pilots should be designed in such a way to enable all synchronization and signalisation needed for WPNC PHY layer processing. Separation procedure of pilot signals from payload have to be generally done without payload codeword decoding process, which should be realized only at the final destination. This property is specific to WPNC. Reliable pilots must use enough "energy" to provide its function even in condition which doesn't guarantee correct payload codeword decoding, but signalisation must be provided.

Acquisition Pilot (PiAcq)

This pilot serves for identification of the node's activity. It means that every node have to have assigned unique "signature" in form of PiAcq. The second purpose of PiAcq is synchronization of slots/packets which corresponds to the transmitters activity stage slot. In other words slot synchronization means the alignment of the packet inside the slot.

Hierarchy Pilot (PiHrc)

This pilot should provide signalistation of graph fragments of predecessor nodes, in case of more advance network scenarios, directly at PHY layer. It should contain information about hierarchical functions of the node itself and all preceding nodes as well as information about network topology. This feature is necessary for more complex wireless networks. It also should provide information about relay operations and ID of HNC map. This pilot signal is very specific to HDF relaying strategy.

Channel State Estimation Pilot (PiCSE)

PiCSE should serve for node-local CSI estimator. However in case of multisource setup there will be channel parametrisation with higher complexity. The WPNC decoding processes are strongly dependent on the relative channel coefficients. Therefore the CSE pilots must be designed in such a way to be resistant to the non-orthogonal superposition while still providing a high quality relative channel fading coefficient estimates.

3.2.2 Implemented protocol

In this subsection is defined interface protocol with respects of desired functionality. To simplify our situation in this solution of protocol are not considered hierarchical or CSE pilot signals, because it is beyond scope of this work. We

can also neglect all stage scheduling processes which arise in advance ad-hoc dynamic networks due to a priori known network topology and cloud scenario. In order to simplify situation even more, we can assume frame synchronization and symbol timing due to synchronous transmitters/receivers respectively. But this is not completely correct statement for reasons shown in 3.4.2. We can also neglect time of signal propagation due to short distances between Tx/Rx. So there is no need of any timing advance based methods.

Functionality of interface protocol like this should be sufficient enough to provide communication in wireless network scenarios used in this framework with respect to half-duplex constraint. The structure of frame for this framework is shown in figure 3.3.

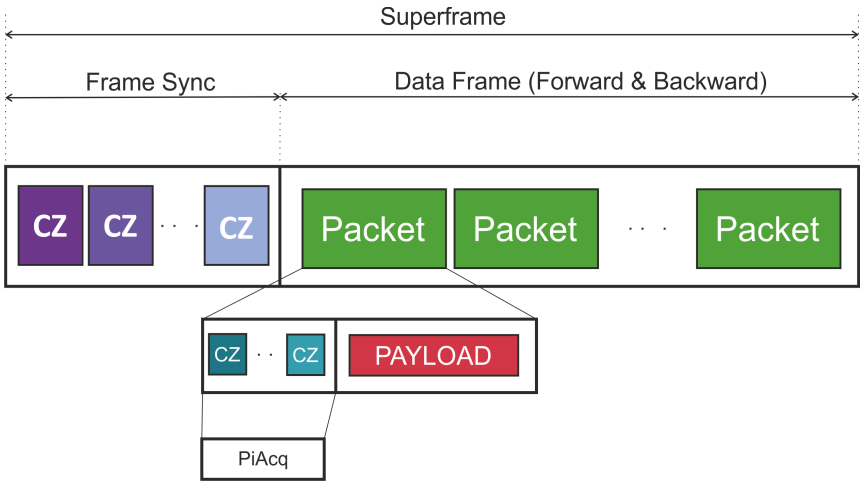

Figure 3.3: Designed frame structure

It can be seen, that beginning of each superframe is signalised with Frame Sync pilot signals, for this particular case were choosen CAZAC codes. Each node in the network has given unique CAZAC sequence, which is known by all other network's nodes. CAZAC codes are thoroughly described in section 3.3. For packet signalisation can be used CAZAC codes or gold codes in this framework. Packet contain superposition of unique informations from every transmitting node. The creation of the payload before modulation or coding processes is following:

Algoritmus 3.1 Payload design algorithm

```
tx1_payload = zeros(symbols_per_packet, superframe_length);
   for i = 0:superframe_length-1
       tx1_payload(:,i+1) =
     data_input_Tx1((i-1)*symbols_per_packet+1:i*symbols_per_packet)];
   end
end
```

This unique data inputs are then modulated, interpolated and after that is being made complex envelope. General linear modulation can be descripted as

$$s(t) = \sum_n g\left(d_n,\, \sigma_n,\, t - nT_s\right) = \sum_n g\left(q_n,\, n - T_s\right) \tag{3.3}$$

where g is modulation function, d_n are data symbols $d_n \in \left\{q^{(i)}\right\}_{i=0}^{M_d-1}$ and σ_n are inside states of modulator (memory of modulation, state equation $\sigma_{n+1} = \sigma\left(d_n, \sigma_n\right)$ define evolution of modulator in time). In general $\sigma_n \in \left\{\sigma^{(i)}\right\}_{i=0}^{M_q-1}$where M_q is number of modulator states. T_s is symbol period.

Next operation in this process is upsampling of modulated signal. Input signal X is upsampled by inserting $N-1$ zeros between input samples. After that is needed to filter upsampled data by a modulation pulse and this data reshape into superframe structure to keep designed protocol rule.

Implementation of these routines is shown in following algorithm.

Algoritmus 3.2 Modulation, upsample and complex envelope implementation

```
% modulator object
   M=tx1_Mod;
   hMod = modem.qammod(M);
% modulation
   tx1_symbols = modulate(hMod,tx1_payload);
% interpolation
   tx1_upsample = upsample(tx1_symbols, samplespersymbol);
% creation of complex envelope divided into two steps
   step1=filter(REC_filter,1,tx_upsample_vectorized);
  tx1_CE = reshape(step1,symbols_per_packet*nSamp/log2(M),SF_length);
```

Following algorithm shows addition of unmodulated CAZAC sequences to a upsampled generarlly modulated data inputs now in form of complex envelope. The constant $k \in \{0.25,\, 1\}$ used to mupltiply amplitude of created complex envelope is standard procedure to improve ability to detect CAZAC sequences even in condition with low SNR ratio. This structure now corresponds with figure 3.3 and this superframe is prepared for transmition.

Algoritmus 3.3 Frame and Packet Sync addition

```
for i = 0:superframe_length-1
   if i == 0
      Superframe1(:,i+1) = [FrameSync1;k*tx1_CE(:,i+1)];
   else
      Superframe1(:,i+1) = [PacketSync1;k*tx1_CE(:,i+1)];
   end
end
```

3.3 CAZAC codes

This part is focused on CAZAC codes and its properties suitable for purposes of frame and packet sycnhronization. Constant Amplitude Zero Auto-Correlation code (CAZAC) is complex-valued periodic signal with modulus and out-of-phase periodic auto-correlation equal to zero. This code is generated by cyclically shifting the basic CAZAC code of a choosen length and as the name of this code prompts this shifted CAZAC codes are always orthogonal to its base CAZAC code.

CAZAC sequences allow precise estimation of the frame-start position in time and also in frequency domain, which is very convenient for our purpose. CAZAC sequences are defined as a samples of a complex exponential function wich can be seen from following formula

$$x_{N,r}(n) = \begin{cases} exp(-j\frac{r\pi n^2}{N}) & for\, N\, even \\ exp(-j\frac{r\pi n(n+1)}{N}) & for\, N\, odd \end{cases} \tag{3.4}$$

where $x_{N,r}(n)$ denotes a sequences of length N and root of r. This root have to be prime number for ability to provide following properties.

1. CAZAC sequences are periodic with period N if N is odd. It is given by

$$x(n+N) = x(n)$$

2. Auto correlation function of CAZAC sequence with a cyclically shifted version of itself is zero, it is non zero in one and only case when this shift corresponds to the cyclic shift (length of the sequence). It can be seen on following figure 3.4 that CAZAC sequence with $r = 13$ and $N = 127$ is the same sequence like sequence with $r = 140$ and $N = 127$.

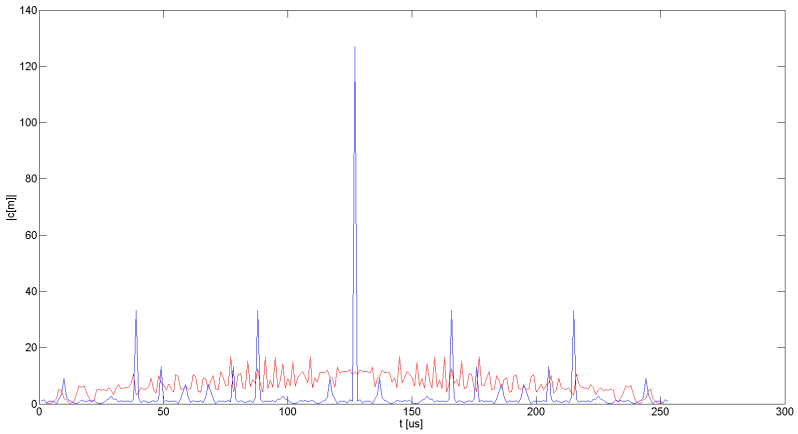

Figure 3.4: Cross correlation of CAZAC sequence shifted of 1 periode

3. The cross correlation between two CAZAC sequences with different root values r is always constant $\frac{1}{\sqrt{N}}$, see figure 3.5. On this figure are two cross correlation functions. Blue one is convolution of one period of CAZAC sequence with itself, where $N = 127$ and $r = 13$. The red one is cross correlation between two CAZAC sequences with $r_1 = 13$, $r_2 = 17$ and common length $N = 127$. Note that peak at $m = 127$ has an amplitude in maximum at $|c(m)| = 127$.

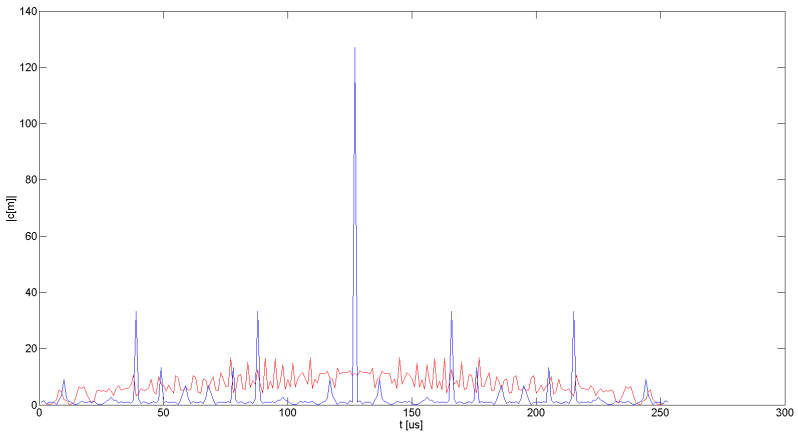

Figure 3.5: Cross correlation of two CAZAC sequences with different roots

It is worth noticing that CAZAC sequences is also very resistant against noise influences. In addition CAZAC sequences have reduced inter-symbol interference, that helps to avoid of interferences between multiple antennas and it also lowers peak-to-average power ratio. Due to this properties, CAZAC sequences find application in wireless communication systems very soon. For example these sequences is used for channel estimation and sycnhronization routines (for instance preamble signature) in long-term evolution (LTE) of 3rd Generation Partnership Project (3GPP). In this framework CAZAC sequences serve for frame and packet synchronization.

3.4 Framework platform

This section is focused on options of possible development environment for this framework platform design. Choosen development tool have to provide comfortable enviroment to satisfy demands on this framework. It means that designer of this framework should be able to build this testbed in such a way to fulfil functionality described in section 3.2.1 and also to provide adequate enviroment for implementation of interface protocol (included design of pilot signals) with

ability to expand this framework to a more complex application. Namely ability to build user friendy graphical user interface to control whole framework and expand this framework for instance to another application with remote access if possible. After considering all of these features there only two meaningful options left.

1. *GNU radio* is open-source software development toolkit that provides signal processing blocks to implement software radios. It provide optimized enviroment for management of Universal Software Radio Peripheral (USRP) for Tx/Rx in wireless network cloud scenarios. It also allows to use external high-accuracy timing reference and distribution system (OctoClock-G) for USRPs to fulfil condition of precise frame synchronization and symbol timing in multi node cloud scenarios. On the other hand GNU radio is open-source development toolkit with lack of official support or documentation.

2. *MATLAB* is a high-level language and interactive environment for numerical computation, visualization, and programming. Matlab is also ideal instrument for algorithm developement with various forms of additional toolboxes (for instance Parallel Computing toolbox, Signal Processing and Communications toolbox etc.). In addition Matlab allows to create advanced Graphical User Interfaces (GUI) to provide a comfortably and user friendly solution how to control created applications or scripts.

Despite of no guaranty of frame synchronization or symbol timing was selected Matlab as a base environment tool to design this framework. It can be problematic but in next sections is an effort of algorithm developement to provide this synchronization demand. In this point of view can be choice of GNU radio more optimal, but to provide perfect synchronization is necessary to buy another expensive device. So the target of next section is to achieve same functionality without spending another financial resources.

Both variants have advantages and disadvantages so the decision about which enviroment select was very difficult. The GNU radio provides a better performance of USRPs control but Matlab allows to expand designed framework to an effective GUI based application. In addition there is also possibility to make a web based application in case of Matlab based platform. But this topic is further descripted in chapter **??**.

3.4.1 Real-time signal processing

Nowadays topology of wireless networks is generally dynamic due to portable devices which are moving in real-time. This is a huge complication for stage scheduling because time delay caused by signal propagation is constantly changing. This means that receiver in these networks can not operate in its relative clock time given by a local oscillator, because there is no guaranty of receiving any useful data. The method how to guarantee a recieving proper data sequence

is based on a pilot signal detector which continously in time buffers and compares recieved signal with defined pilot signal indicating start of transmitted signal. If pilot detector register a start of transmitted signal it gives an order to a memory bank to store a defined number of incomming samples N. This number of samples is strictly given by a protocol in current network. Stored data then represents a useful data message from a transmitter object. The decoding routines have to start immediately after recieving the last symbol of the usefull message. This routine is not so simple as it seems. The memory implementation allows to read and write simultaneously, but have to be prevented a case when second data frame arrives. It can cause a situation when are currently processing data overwritten by a new data message. This is treated by switching banks of memory that stores a whole message. When is one bank of memory filled with data, it is switched with the empty one and whole payload is provided to Digital Signal Processor (DSP) to further processing. The common approach how DSP access data in memory from is realized by direct memory access (DMA). Meanwhile is filled second bank of memory in case of the next packet arrieved. This process in simplified form is depicted in figure 3.6.

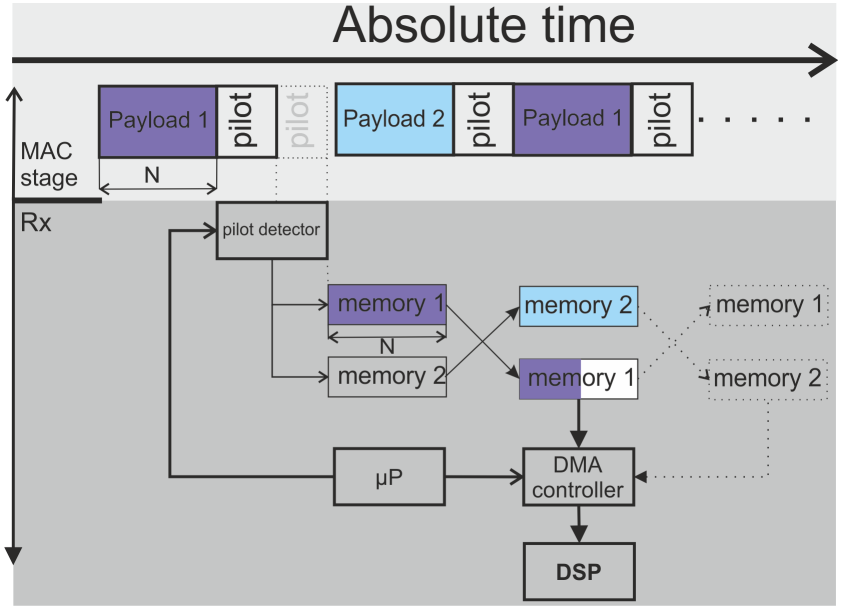

Figure 3.6: Real-time signal processing

This approach is called real time signal processing. It does not mean that processing is operated on every data symbol recieved in given time but simply real time systems have to guarantee response within a strict time constraint often this time is reffered usualy in range from microseconds to miliseconds. In other words real-time system have to recieve data, process them and return the result fast enough to affect the environment at that time.

The term real-time is often confused with other usage of this terms. Namely in the domain of simulations were this term means that simulation's clock runs as fast as a real clock. In real-time based signal processing have to be fulfilled one condition which is bounded processing delay even if the processing continues for a unlimited time. In other words it means that the mean processing time per sample is no greater than the sampling period.

Due to this circumstances real-time signal processing have to be implemented as close as possible to physical layer to satisfy a very quick response. All synchronization routines should be implemented in FPGA of the receiver in order to guarantee a fast enough response. Because developed framework is based on Matlab enviroment, which refers to an application layer, there is not in our capabilities to attach a real-time signal proceessing via Matlab based platform.

Therefore in this work we were forced to step aside from the real-time model and the real-time sycnhronization of Rx/Tx processes have to be reached otherwise. Next sections are focused on routines developement to provide a strict sycnhronization of devices in network. It should lead to a synchronized Tx/Rx processes in order to guaranty of fully received frame segments. This approach is not quite exact but it should be a sufficient for defining basic cloud scenarios in order to developement and testing of synchronization routines such as CSE and others.

3.4.2 Wireless cloud implementation

On the beginning of the framework development there were a questionable possible solutions of process synchronization which should guarantee a perfect symbol timing and frame synchronization. This feature should substitude a real-time synchronization of Tx/Rx processes. With consideration of all SW potentials we decided that there are only three ways how to create a wireless network with multiple nodes simultaneously communicating. This is in general quite problem, because every program written in conventional way is executed command after command, in another words serially-executed. To provide simultaneous communication of multiple (in this case three) devices, it's neccessary to find a way how to parallelize this process. Very important feature of this task is the realization of network scenario physically built in laboratory. Scheme of this network is depicted in figure 3.7.

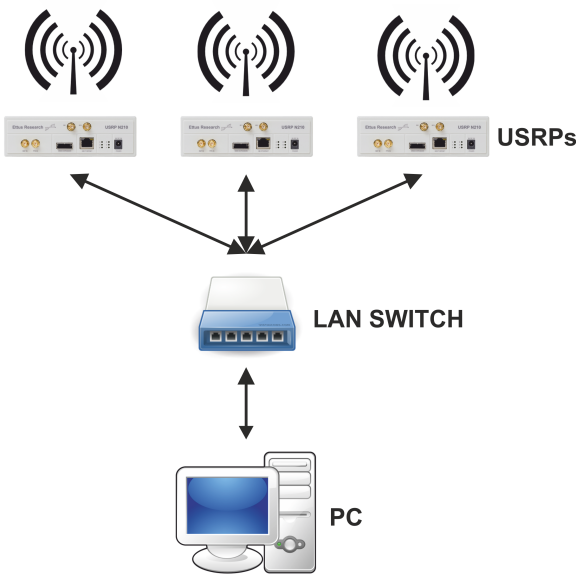

Figure 3.7: Network scheme

Figure 3.7 shows that communication between PC (with invoked Matlab instance) and individual USRPs is via LAN switch, that also causes unpredictable long delay in the information path. But in mean this delay can be considered as constant and uniformlly distributed to all USRPs. But in general this network scheme degrades any synchronization procedures created in application layer. In order to keep precise and time invariant synchronization of multiple devices, it is necessary to implement this process directly on PHY layer. Unfortunatelly Matlab does not provide a coordination of processes by an external precise clock source. Therefore following sections are focused on approaches how to reach the same result with curent state-of-art resurces.

There is three general approaches how to fulfil condition of frame synchronization and symbol timing for three devices simultaneously in Matlab based framework platform:

1. *Sequential programming:* This approach in its nature can never fulfil the condition of simultaneous control of three devices. The main idea of this approach is that time difference between two transmitters is given only by one command execution in Matlab. The question is if this method will be sufficient enough for this framework or if is needed another solution of device synchronization. Implementation of this approach is thoroughly descripted in subsection 3.4.3.

2. *Parallel Computing:* Matlab enviroment includes Parallel Computing Toolbox. This toolbox provide ability to solve computationally and data-intensive problems using multicore processors. This parallelization is de-

signed to offer boost in data processes. It is effective especialy with array operations and parallelized numerical algorithms. Athough this toolbox is not developed for synchronization of peripheral devices as USRPs, in section 3.4.4 we try to design algorithm in such a way to be able provide synchronnous operational mode of active devices.

3. *Invoke multiple Matlab instances:* This method can be implemented but it suffers with lack of frame and symbol synchronization. In basic form of this method is guaranted only simultaneous communication of multiple nodes (based on how many instances of matlab are invoked). There is possible improvement of this method to satisfy demand of frame synchronization and symbol timing by a clock counter routine with higher priority than matlab instances. If this "clock" allows to send command to all matlab instances simultaneously, it will provide frame and symbol timing synchronization.

3.4.3 Sequential programming implementation

This approach has very strightforward advantage in form of implementation simplicity. But on the other hand it suffers with lack of ability to fulfil a key requierement for this framework platform i.e. synchronous Tx/Rx operational mode. This is crucial disadvantage of this method. The question is if the delay between two transmitting processes will be long enough to disable usage of HNC decoding processes. This inserted delay between transmissions is given by one command execution in Matlab. Theoretically it can be processed in such a way, that the machine cycles of matlab will have almost zero impact of synchronization error. But this is only theoretical assumption, in reality is this delay caused by one machine cycle shown in figure 3.9.

The instruction flow chart for this approach of sequential algorithm is depicted in figure 3.8.

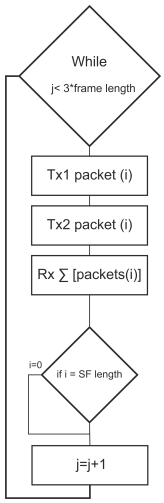

Figure 3.8: Instruction chart flow of sequential algorithm

Implementation of this algorithm into Matlab can be done with different ways via different cycles. In algorithm 3.4 is implemented algorithm with cycle *while* which stops after super frame is three times send. This makes a framework more robust and resistible against unpredictable enviroment influences.

Algoritmus 3.4 Sequential algorithm implementation

```
frameLength=180;
R=zeros(frameLength,3*SuperFrame length); %preparation matrix for received
data
i = 1; j = 1; k=1; %initialization of cycle indexes
nStop = 3*SuperFrame length;
while(j<nStop)
   step(hTx1,final1(:,i)); %Transmission of packet(i) for Tx1
   step(hTx2,final2(:,i)); %Transmission of packet(i) for Tx2
   [Y, LEN]=step(hRx); % Recieved superposed packet(i) from Tx1 and Tx2
   data=Y;
      if mod(i,nPacket)==0 %end of super frame, start again with packet 1
         i = 0;
      end
      if LEN==frameLength
         R(:,k)=data;
         k=k+1
         if k == nStop
            break;
         end
      end
i = i+1; j = j+1
end
```

We are able to distinguish start of packets from each transmitters only if we know root number assigned to each transceiver at destination point. This number have to be unique for each Tx, as mentioned in section 3.3. Lets assume that samples of received signal are$x[m]$ and *a priori* known CAZAC sequence for $Tx1$ is $cz_1[m]$. In order to find a start of packet $Tx1$ we apply cross correlation function 3.5 to find the beginning of frame sent from $Tx1$. The physical meaning of this function is comparing a mutual energy of two sequences. If is the one sequence contained in second one (it can be shifted) it causes a maximum of cross correlation function. Position of this maximum is given by a mutual shift of these sequences.

$$(x \star cz_1)[n] = \sum_{m=-\infty}^{\infty} x^*[m]\, cz_1[m+n] \tag{3.5}$$

Analogically we can get starts of packets from $Tx2$. In order to demonstrate this delay between transmissions, both cross correlation functions are depicted into one figure 3.9 where could be seen synchronization inaccuracy. In figure 3.9 overleaping blue and red peaks represents a signalised beginnings of packets from each source. It is also noticible that red peak is slightly ahead of blue peak from time flow point of view. Other two blue side peaks are caused by non optimal select of root of CAZAC code.

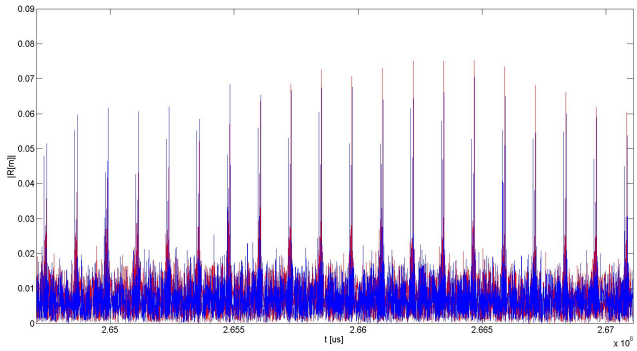

Figure 3.9: Synchronization inaccuracy for simultaneous 2Tx-Rx

In order to further analyzation of this time interval between transmissions is provided zoomed figure of the same scenario and received signal in figure 3.10.

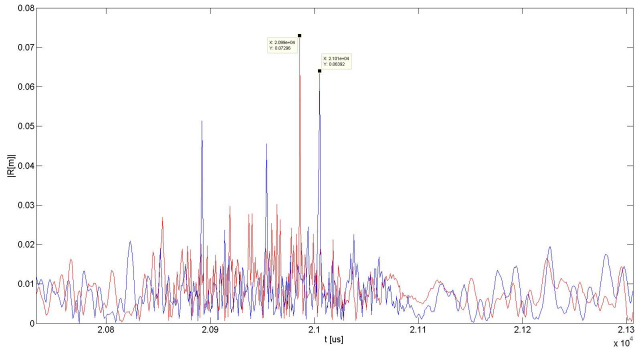

Figure 3.10: Zoomed frame synchronization slip

This method is not entirely exact, but has its own advantages. Besides already mentioned implementation simplicity, with this approach can be designed all network cloud scenarios for this framework. The synchronization slip is unavoidable in this method but is stabile so future designer of synchronization and decoding routines can involve this delay to his algorithms.

3.4.4 Parallel computing implementation

The days when was main target of processor design its frequency are gone. Multicore processors are the new direction manufacturers are focusing on. Using multiple cores on a single chip is advantageous in raw processing performance with ability to handle more processes simultaneously. This trend lasts over

five years and brings new opportunities and solutions to program designers. And multicore processing is a key feature in this method. We try to use all resources of Matlab enviroment to design algorithm in such a way, to achieve synchronization of all devices used for this task.

Matlab Parallel Computing Toolbox can split execution of program on parallel parts using multicore processors. This toolbox let us use the full processing power of multicore processors by executing applications on workers (Matlab computational engines) that run locally.

With this toolbox we can theoretically parallelize two or more (based on number of cores of particular PC) by a creating multiple Matlab workers which should operate indenpendently of each other. This method is to the eye very sofisticated and easy to implement. But the parallel routines provided in this toolbox does not allow to user assign commands to individual cores. This unability to allocation of individual processes to workers can be crucial and in worst case it can makes this method unrealizable.

The first and most easy way how to discover behaviour of parallelization of Matlab is to rewrite algorithm 3.4 to a parallel for loop provided by a Parallel Computing Toolbox. Assumption that just a change of loop cycle type for parallel one is very naive, but it is good start point of algorithm developement. The chart flow of this basic algorithm is shown in the figure 3.11.

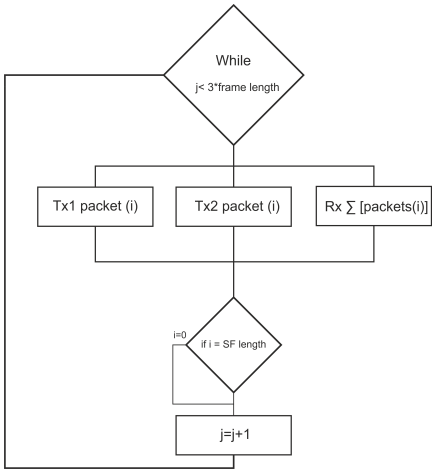

Figure 3.11: Basic parallel algorithm chart flow

Implementation of this algorithm is basicly the same like algorithm 3.4. During testing procedure of this algorithm error immediately occurs. It was caused by worker collision on *step* instruction which reserves a USRP object for a short period of time and multiple workers try to communicate with the same USRP. And because there is no rule or authority to avoid this collision, this method is impracticable.

One of the routines which should avoid this collision is based on partition of

transmision routines into individual functions. This can be done by a function array, which will be processed within parallel loop. The chart flow of this algorithm is shown in figure 3.12.

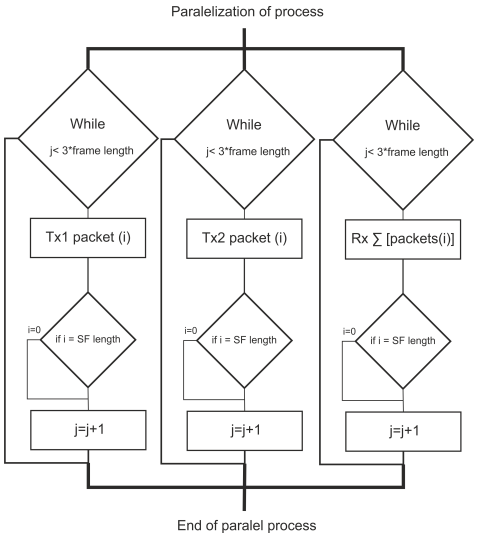

Figure 3.12: Sliced parallel algorithm for collision avoidance

This algorithm in its nature does not provide a fully synchronization of processes but only parallelization of transmission routines. But it is only way how to check if the routine for allocating workers for each called function is working or not. It was done by replacing step functions by an easy display functions. It was proven that each cycle there were three displayed messages in matlab workspace. After this testing phase was algorithm implementd in Matlab as shows following algorithm.

Algoritmus 3.5 Sliced parallel algorithm

```
matlabpool (number_of_devices);
funList=(@func1,@func2,@func3);
i=1;j=1;k=1;
    while(j<nStop)
    parfor l=1:length(funList)
        funList{l}(final1,final2,hTx1,hTx2,hRx);
end
    if mod(i,nPacket)==0
        i = 0;
    end
data=Y;
    if LEN==frameLength
        R(:,k)=data;
        k=k+1
    end
i=i+1,j=j+1,k=k+1;
    end
% example of func1
function func1(final1,~,hTx1,~)
step(hTx1,final1(:,i));
end
```

We have to send variables and objects to every function as an arguments because called functions have a independent workspaces. Unfortunatelly this algorithm has a one crucial problem i.e. initialization of USRPs every loop cycle which causes ridiculously long delay between each cycle. This can be solve by creation of the same loops within the each function. But anyway this parallelization also does not guarantee an sycnhronization of the parallel cycles. These features should be treated in next algorithm adjustment. The lack of synchronization of these parallel loops will be treated by a barrier routine. Because we cant use a Matlab barrier routine provided by a *spmd* function in form of *labBarrier* command, we have to find out another solution of parallel tasks sycnhronization. For this purpose is used a semaphore routine. Semaphore is program written in *C* to provide semaphore routine in Matlab enviroment by compiling this *C* program into a *MEX* file. This semaphore works on relative easy principle and manipulation with semaphore is similar to function. In following algorithm will be described semaphore functionality.

Algoritmus 3.6 Semaphore functionality

```
semaphore('create',uniquekey,init_value)
    % for example uniquekey=3; init_value=2;
semaphore('wait',uniquekey)
    % init value decremented by one
semaphore('wait',uniquekey)
    % init value decremented by one (0 - locked)
semaphore('post',uniquekey)
    % init value incremented by one
semaphore('post',uniquekey)
    % init value incremented by one - unlolck processes
```

The algorithm charflow with used semaphores to synchronize all transmission routines is depicted in following figure 3.13.

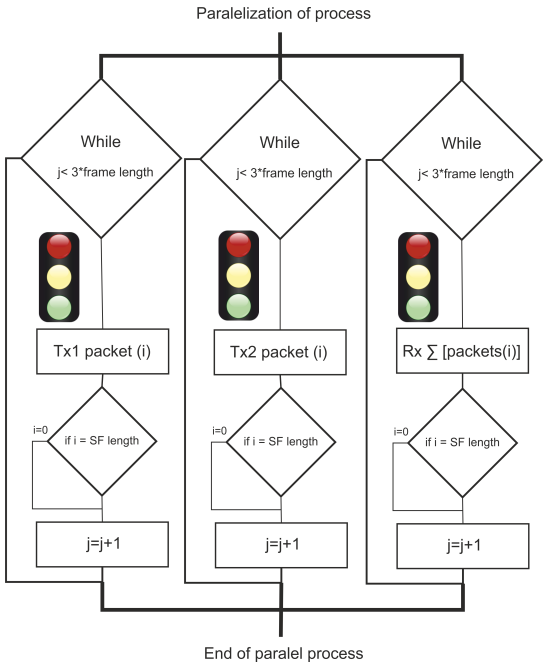

Figure 3.13: Parallel algorithm with semaphores

This algorithm have to use extra one loop to control semaphores which are used to create a barrier right before each *step* function. Therefore is key feature of this algorithm *barrier function* which create a barrier for the first three workers accessed this function till the forth one unlocks all these workers at once. Implemented barrier function is shown in following algorithm 3.7.

Algoritmus 3.7 Barrier function

```
function barrier()
semaphore('wait', cntMutexKey); % separation of parallel tasks
barrierCnt = barrierCnt + 1;
  if(barrierCnt == 4) %We now know that func1,func2,func3 are waiting
    barrierCnt = 0; %reset count
    semaphore('post', cntMutexKey);
     semaphore('post', barrierKey); %Increment barrier count, so a func will
run.
     semaphore('post', barrierKey); %Increment barrier count, so a func will
run.
     semaphore('post', barrierKey); %Increment barrier count, so a func will
run.
  else
    semaphore('post', cntMutexKey);
    semaphore('wait', barrierKey); %Wait for other threads (this is a barrier).
  end
end
```

Unfortunatelly this algorithm isn't realizable due to unability to set a counter of *barrierCnt*. It is necessary to pass this variable via an argument to this function so every worker who access this function sets this variable again to value 1. Therefore is necessary to find another solution then via *parfor* loop. This functionality could be provided via function *spmd* wich allows to execute the body of this function in parallel. In addition it should provide more sophisticated allocating of tasks to each worker. In following algorithm we divide workers by special *labindex* command which is provided by function *spmd*. This index refers to a unique number of parallel worker created in Matlab. Therefore it is very convenient tool how to separate tasks via switch-case nesting function. At top of that, the body of *spmd* can work with base workspace therefore is no need to passing variables via arguments. The following algorithm represents implementation into Matlab.

Algoritmus 3.8 Adjusted parallel algorithm with barrier

```
barrierKey = 4; % initialize value of barrier semaphore
semaphore('create', barrierKey, 3); % has count of 3 (devices) +1 to controll
semaphore('wait', barrierKey); % 2
semaphore('wait', barrierKey); % 1
semaphore('wait', barrierKey); % now it has 0 (next 'wait' is barrier)
matlabpool(4)
spmd
  switch (labindex)
    case 1
      while(j<nStop)
        semaphore('wait', barrierKey)
        step(hTx1,final1(:,i));
          if mod(i,nPacket)==0
            i = 0;
          end
        i = i+1;j = j+1;
      end
    case 2
      while(j<nStop)
        semaphore('wait', barrierKey)
        step(hTx2,final2(:,i));
          if mod(i,nPacket)==0
            i = 0;
          end
          i = i+1;j = j+1;
      end
    case 3
      while(j<nStop)
        semaphore('wait', barrierKey)
        [Y, LEN]=step(hRx1);
        data=Y;
          if LEN==frameLength
            R(:,k)=data;
            k=k+1; j = j+1;
          end
        end
    case 4
      while(j<nStop)
      pause(0.1) % protection interval other workers should wait on barrier
      semaphore('post', barrierKey)
      semaphore('post', barrierKey)
      semaphore('post', barrierKey) % unlocks the USRPs
      j=j+1;
      end
    labBarrier;
end
```

During testing of this algorithm several errors occurs randomly. It means that sometimes transmitting process of two transceivers pass without any errors but usualy an error occured. In cases when error occurs, it was due to unability of Matlab to provide data for USRPs and in couple of cases again due to an unability to reach USRP in particular time of error. The cause of this behaviour may be the fact, that in spite of three/four matlab parallel workers were created, it does not guarantee an physical allocating of processor cores to each worker. This schedulling is strictly in control of OS scheduler.

The error where is not supported data for ettus had a key part *"requested number of samples. Expected 3597, Found 0"* that can be caused by two features. The first one is the unability of Matlab to give a high amount of data to all USRPs at once. But it is highly unlikely. Therefore we were also testing the same algorithm with half packet length and after that with quarter of packet length. But the result was be the same. The second variant can be limitation of current state-of-art network scheme depicted in figure 3.7 because it could be caused by a limitation from an IP packet structure, where is the MTU (Maximum Transmission Unit) for Ethernet defined as 1500 bytes per instance. That could be solved by a workstation equipped with three ethernet cards directly connected to each USRP.

3.4.5 Multiple Matlab sessions

This approach is based on invoking multiple Matlab sessions. The transmission routines are wholly independent and in this approach is guaranted avoidance of Matlab collision in communication with USRP. The principal scheme of this approach is depicted in figure 3.14.

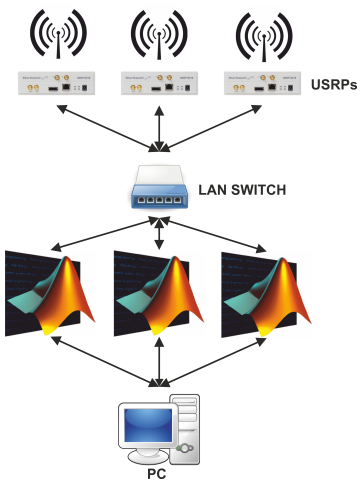

Figure 3.14: Multiple Matlab sessions

The huge disadvantage of this method is lack of any sycnhronization of these

processes. Therefore is no guaranty of recieving an usefull data. But it can be compensated by a very long cycle where is multiple times transmitted the same superframe. This method is realizable but not optimal one.

Due to these circumstances the frame synchronization and symbol timing seems to be unrealizable via the Matlab platform. However, the sequential algorithm were not provide so bad results and therefore is the framework platform based on this algorithm. It is sufficient enough to provide tool for a design of synchronization algorithms and other experiments.

Chapter 4

Framework implementation with GUI

Main goal of this chapter is to develope a Graphical User Interface (GUI) for framework. We will descibe two basic approaches how to buit GUI in Matlab with examples. We also try to mention basic Matlab graphical control units and describe hierarchy of Matlab graphical structure. This hierarchical structure is key factor for understand data flows between multiple GUIs and scripts. Final product of this chapter should be creation of stand-alone application which provide full functionality and also brings portability of this framework to another devices. In addition, functionality of this application will sustain even on devices without Matlab installation. This fact had relevant role during decision making about suitable platform for this framework.

When is program used by a several persons the raw script is can be confusing for those which have knowledge about used language.

4.1 Approaches of GUI development

A graphical objects were already mentioned but these objects weren't defined yet. Graphical object is every basic elements used for displaying any kind of graphical output. For clearly understanding core of this topic is very important to know mutual organization and subordination of these objects - hierarchy. Matlab graphical objects hierarchy is shown in following picture4.1 and is very important for data management in more advanced GUIs.

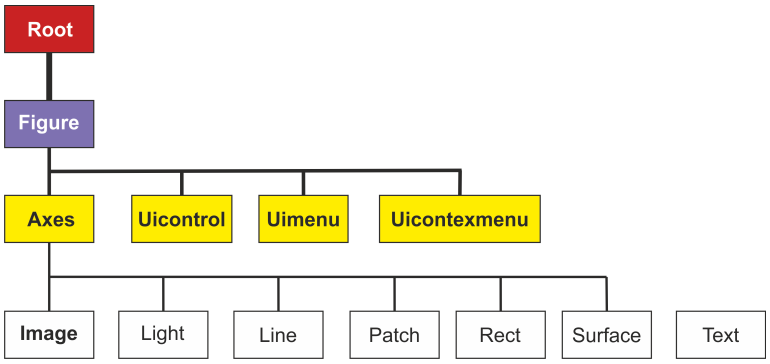

Figure 4.1: Hierarchy of basic graphical objects of Matlab system

A Matlab GUI is a figure window to which designer can add user-operated graphical components. Designer of GUI have to select type, size and position of graphical components as he likes, but generaly it should be designed in such a way to provide well-arranged tool for end users. The functionality of any graphical component is given by a property called *Callback*. Using callbacks allows the components do its function by only clicking on that object.

System Handle Graphics was implemented to Matlab to provide a effective control of all graphical objects. The main feature of Handle Graphic system is a unique number assigned to every graphical object in Matlab.

In figure 4.1 is depicted objects hierarchy and if some graphical object is created then is uniquely determined by a handle value. Except of the tobject *Root* at a top of hierarchy which has strictly given handle value *0*. For example we create a new graphical object figure by a command *h=figure* and the following figure object appears.

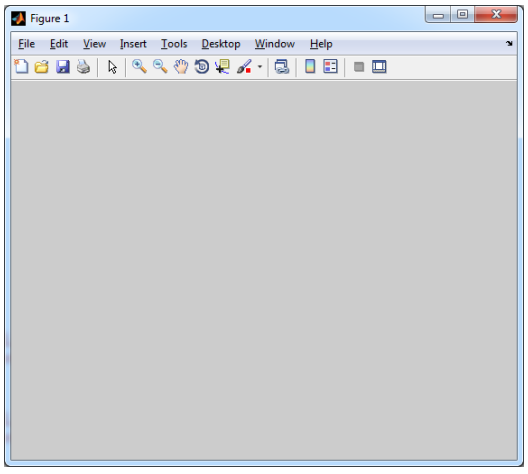

Figure 4.2: Figure object example

By command *h=figure* was unique handle number assigned to variable h wich helps with further work with figure. Now we can check value of variable h and the result is $h = 1$. In other words *Handle* of object *Figure* in this case is number *1*. We can imagine figure object like a window with some name (Figure 1) which is ready to define underling graphical objects (*Uicontrol, Uimenu, Uicontexmenu*) in Matlab hierarchy depicted in figure 4.1.

Another important feature of GUI is a fact that every GUI has its own workspace independent of other worskpaces where are stored variables and objects in form of local variables. For instance if we declare variable or object in GUI, these objects are invisible to Matlab's base workspace and vice versa. This feature brings several obstacles which will be described in following text.

There are only two basic approaches of developing GUI in Matlab:

1. *Programmatic GUI construction:* created code files generates GUIs as a functions or scripts.

2. *GUIDE (GUI Developement Enviroment):* is an interactive GUI constuction kit.

The code files of these two approaches look different. Programatic GUI files are generally longer, because they explicitly define every property of the figure and its controls (position, visibility, name, etc.), as well as the callbacks for these objects. GUIDE GUIs define most of the properties of objects within the figure itself. All these properties are stored in its FIG-file instead of directly in its code file. The code file then contains only callbacks and other functions that initialize the GUI when it opens. If is GUI builded via GUIDE there is possibility to modify it programmatically. However GUI created programmatically can not be then modified with GUIDE. Ways how to build a GUI is shown in following text.

4.1.1 Programmatic GUI construction

Using this approach, designer creates a code file that defines all component properties and behaviors. When a user executes the file, it creates a figure, populates it with components, and handles user interactions. It can be demonstrated on following example where will be created a main menu similar to main menu of framework created later in this work. It can be done by following algorithm.

Algoritmus 4.1 Programmatic main menu example

```
   function example()
% initialization part
   f = figure('Visible','off','Position',[360,500,450,285]);
      hcloud = uicontrol('Parent',f,'Style','pushbutton','String','Cloud',...
      'Visible','on', 'Position',[170,220,80,25]);
   hTx = uicontrol('Parent',f,'Style','pushbutton','String',...
      'Transceivers','Position',[170,190,80,25]);
   hsync = uicontrol('Parent',f,'Style','pushbutton',...
      'String','Sync','Position',[170,160,80,25]);
   hR_strat = uicontrol('Parent',f,'Style','pushbutton',...
      'String','Relay Strategy', 'Position',[170,130,80,25]);
   hrelay = uicontrol('Parent',f,'Style','pushbutton',...
      'String','Relay', 'Position',[170,100,80,25]);
   hreceiver = uicontrol('Parent',f,'Style','pushbutton',...
      'String','Receiver','Position',[170,70,80,25]);
   hSim = uicontrol('Parent',f,'Style','pushbutton','String','Simulation', ...
      'Position',[170,40,80,25]);
   align([hcloud,hTx,hsync,hR_strat,hrelay,hreceiver,hSim],'Center','None');
% Make the GUI visible.
   set(f,'Visible','on') end
```

This code creates a figure with default *uicontextmenu* and *uimenu* and several *pushbuttons* as shows figure 4.3.

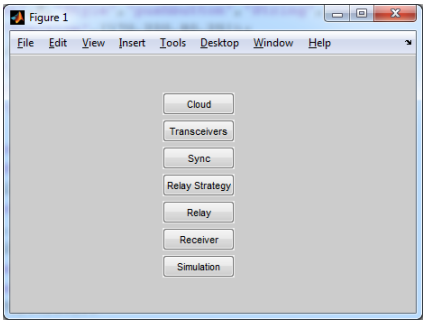

Figure 4.3: Raw main menu example

This GUI has no functionality yet because there are no callback functions defined. Also name is default and objects uimenu and uicontextmenu are irrelevant here. Next algorithm provide change name of figure, define callback function of *Cloud* pushbutton and also hide uimenu and uicontextmenu.

Algoritmus 4.2 Callback function and adjustment of figure

```
% uimenu and ui contextmenu removal
   set(f,'Name','Main menu','MenuBar','none');
% definition of callback function to object property
   set(hcloud,'Callback',{@cloudbutton_Callback});
% execution of callback function
   function cloudbutton_Callback(source,eventdata)
      f2=figure('Visible','on','Position',[800,400,600,600], ...
      'MenuBar','none','Name','Cloud');
   end
```

The *Cloud* pushbutton callback function was defined to invoke another figure object. Demonstration result of GUI and pushbutton behaviour with invoked second figure object is shown in figure 4.4.

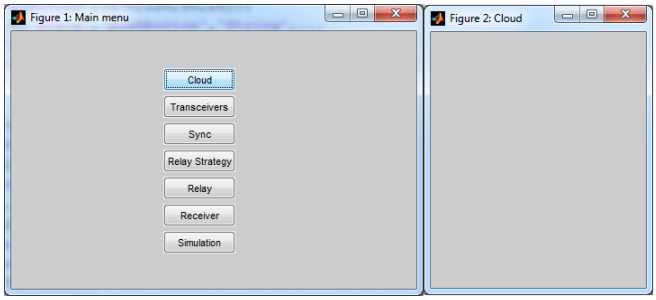

Figure 4.4: Callback function execution and figure adjustment

This is simple demonstration of programmatic GUI developement in Matlab. Advantage is that designer define only properties and callback function which is necessary for desired functionality of GUI, but on the other hand, in more complicated and advanced GUIs with multiple windows, created codes is very large. Due to this feature orieantation in this code is then more difficult. Hence is prefered way to build a GUI with using GUIDE.

4.1.2 Graphical user interface development environment

Matlab have a tool designed especially for developement of GUIs and it is called GUIDE. Key feature of GUIDE is a graphic layout editor, where is created figure and a source code is then automaticly generated.

This approach starts with a figure that can be populated with components from within a graphic layout editor. GUIDE creates an associated raw source code file containing callback functions for the GUI and its components. GUIDE saves both the figure (as a FIG-file) and the code file. Opening either one also opens the other to run the GUI. Navigation in this approach is very easy, only by clicking in layout editor on desired graphical object, it redirects on the callback

function of the same object in source code file. In spite of little complicated file structure in form of two files on every GUI, is this approach easy manageable.

In previous section were mentioned that GUIDE approach was prefered to design GUI for framework testbed. So for the demonstration of this method is choosen example how to create main menu for this framework. In section 4.1.1 we build a main menu programmatically now we build the same one with GUIDE.

Seven *pushbuttons* were putted in figure represented in form of layout editor with equal distribution to fulfil design norm as it shown in figure 4.5.

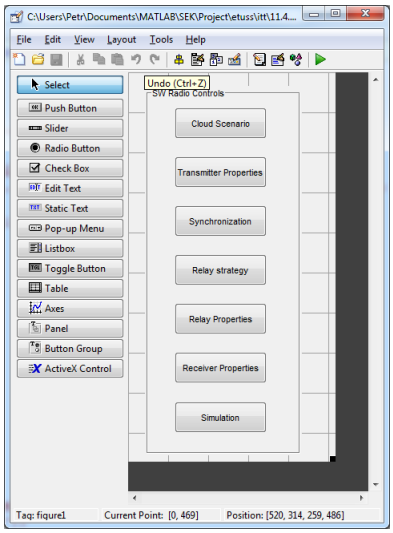

Figure 4.5: Main menu with GUIDE

After save this *fig.* file it automatically generate prepared raw source code with callbacks functions of all graphical objects in *fig.* file. But the structure of automatically generated GUI code file is more complicated. Besides callback functions it contain core, initialize and output functions. Core function is used to association with *fig.* file designed in layout editor and to define singleton. Singleton means that this particular GUI can be invoked only once in contrast with programmatic GUI were defaultly created figure can be invoked multiple times.

Initialize function serve for initialization of graphical components or other variables before the GUI turns to visible. It is useful when some graphical objects have variable base properties dependent on other GUI for instance. In this work is initialize function used to check already defined properties for framework platform.

Main menu serve for navigation in framework platform. It designed in such a way, to provide all settings necessary to run simulation. From this menu is invoked another GUI that serves for particular purposes mentioned in following

sections. Algorithm 4.3. shows implementation of callback function generated by GUIDE. This function opens another GUI namely Cloud GUI which provide selection of desired cloud scenario.

Algoritmus 4.3 Callback function in GUIDE

```
% — Executes on button press in pushbutton_mm_cloud_scenario.
    function pushbutton_mm_cloud_scenario_Callback(hObject, eventdata,
handles)
% hObject handle to pushbutton_mm_cloud_scenario (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
   Cloud; %Calling GUI Cloud
```

4.2 Framework GUI

In previous section was designed main menu of this framework platform. This menu serves for coordination of all GUIs created and to store data from these GUIs. In other words main menu in this platform only invokes other GUIs which serve for particular function. But main menu in this case has no influences on the final simulation. Framework simplified structure with data flows between individual GUIs is shown in following picture 4.6 .

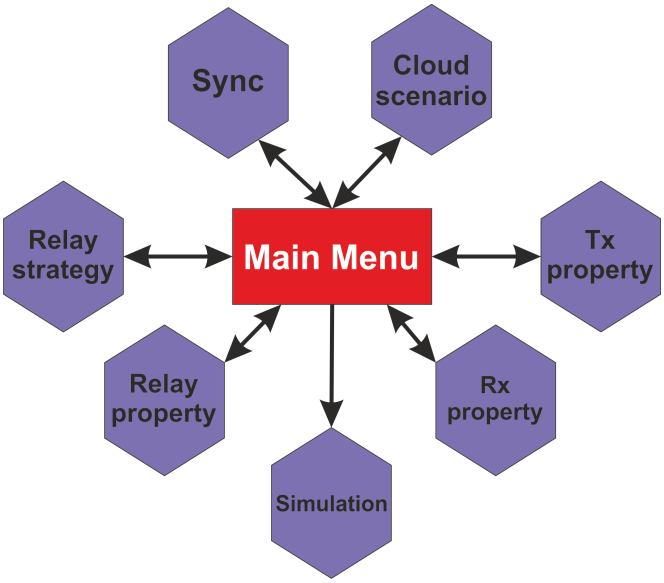

Figure 4.6: Framework GUI core structure with data flows

This framework is designed in such a way that user have a various options how to configure his desired wireless network cloud.

It provide:

- Variable wireless network scenarios.
- Variable Tx properties (gain, frequency, offset, etc) included variable payload for transmission.
- Adjustable SuperFrame structure namely lenght of packets or SuperFrame length.
- User can select one of two supported modulations and also one of the two modulation pulses.
- Relaying strategies
- Adjustable relay properties included file name of recieved signal to be saved.
- Variable Rx properties
- User can select and choose various synchronization options.

4.2.1 Data management in multiwindow GUI

The main issue of data management between individual GUIs is the fact that every GUI has its own workspace where are stored variables and objects in a form of local variables. This makes a data sharing a little bit complicated. There are two approaches how to store data from GUI and after then load this data in another GUI.

1. *Via guidata(hObject,handles):* This method is based on handle structure of GUI. The main disadvantage of this way of storing data is a ability to managa only one variable at any time because another call *guidata (hObject,handles)* overwrites the previously created version of GUI *data.* Therefore is this variant higly non optimal and is not implemented in any part of GUI.
2. *Via setappdata(h,name,value):* Setappdata stores values of *val* in a GUI into a handle of a component *h.* The string of charakters *name* (second argument in ' ') is name for the data. Stored data in this way are uniquelly identified. The value of *h* identify data location and *name* identify a name of variable with stored value *val.* This data can be retrieved any time by using command *getappdata(h,'name').* This method of data management in GUI is much more straightforward and is globally recommended for GUI implementation. In this work are all variables obtained from edit_text boxes or pop_up menus stored in this way.

4.2.2 Network Cloud Scenario GUI

This GUI serves to definition of desired scenario. User can arbitrarily select topology as he likes but there is constrain in form of limited number of USRPs. Available are three USRPs devices which can be used either like transmitter or like receiver. This is caused because of half-duplex constraint. Cloud scenario GUI design is depicted in figure 4.7.

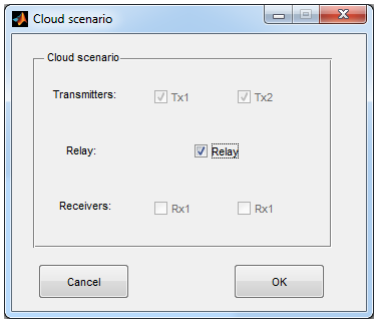

Figure 4.7: Cloud scenario GUI

Figure 4.7 represents a selected scenario with two transmitters and one relay. This GUI is designed with respect to a limited number of devices. When user selects third device, other selections are disabled and can be released by unchecking of the last selected item. Algorithm 4.4 shows how is this routine implemented in Matlab. Principle of this algorithm is implemented in checkboxes callback functions as well but with different device enabled as the last one. The final function of this GUI is to store the number and type of selected devices, to provide initialize functions of other GUI. When the user select *cancel* button, the GUI is simply closed without stored any data and can be invoked again later.

Algoritmus 4.4 Cloud scenario definition

```
function checkbox_Tx1_Callback(hObject, eventdata, handles)
global device
if (get(hObject,'Value') == get(hObject,'Max'))
% Checkbox is checked-take appropriate action
device=device+1;
   if device == 3
set(handles.checkbox_cloud_tx2,'Enable','off')
   **Analogically the rest is disabled
else
set(handles.checkbox_cloud_tx2,'Enable','on')
**Analogically the rest is enabled
end
else
device=device-1;
set(handles.checkbox_cloud_tx1,'Enable','on')
**Analogically the rest is enabled
end
```

4.2.3 Transmitter Settings GUI

In this GUI is very important initializing function, which serves to define pop-up menus string properties. This string is given by result of *findsdru* function which searches for available USRPs in LAN and provide list of IP addresses of available devices. Implementation of initializing function in Matlab is described in algorithm 4.5. This function provide list of all available USRP devices to both *pop-up menus* of this GUI.

Algoritmus 4.5 Initializing function in GUI

```
function transmitters GUI_OpeningFcn(hObject, eventdata, handles, varargin)
set(gcf,'Name','Transmitter settings');
global IP
IP=findsdru;
for i=1:length(IP)
address{i} = IP.IPAddress;
end
address{length(IP.USRPs)+1}='none';
set(handles.popupmenu_tx_sdru1,'String',address);
set(handles.popupmenu_tx_sdru2,'String',address);
```

Next function of this GUI is ability to call another GUIs to define data segments, transmitter properties and super frame options. Structure of this GUI is depicted in figure 4.8.

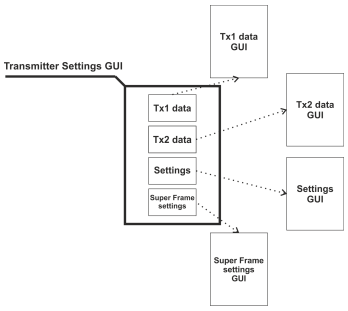

Figure 4.8: Transmitter settings GUI structure

Transmitter settings GUI is designed in such a way to provide explicit definition of IP addresses of USRPs to create transmitter object. It also provide a options to set transmitter properties, superframe properties and data contained in payload of each transmitter device. Design of this GUI is depicted in figure 4.9 where is demonstrated a function of previously created Cloud Scenario GUI. In this case was selected in Cloud Scenarion GUI only one transmitter, therefore is here enabled to modify only one transmitter object as well.

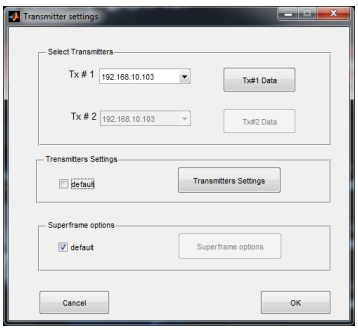

Figure 4.9: Transmitter settings GUI

Transmitter Data GUI This GUI is one of four GUIs that could be invoked by Transmitter settings GUI. The main function of this GUI is to define data to payload of transmitter. Design of this GUI is the same for both transmitters but the data inputs are entirelly separated and independent. This GUI is shown in figure 4.10. Data GUI allows to choose data inputs. It could be one of following options: black and white picture, random data, text message or already modulated signal.

Transmitter Properties GUI This GUI allows to define specific properties of each transmitter (i.e. center frequency, gain, interpolation factor, etc.) and also modulation pulse and modulation. But for safe program run is recommended to use a default properties due to a gain variable. Wrong gain value

can destroy expensive USRP devices. However this possibility is treated by limitation of gain value programatically. All properties that could be change are depicted in figure.

Super frame settings GUI The last GUI that could be called from Transmitter settings GUI allows to modify structure of super frame. Design of this GUI is very simple and is depicted in figure 4.10. It is worth to notice that number of packet value is valid only for random data input. In order to guarantee that all data will be contained in a one superframe it is necessary to adapt this value to size of data input.

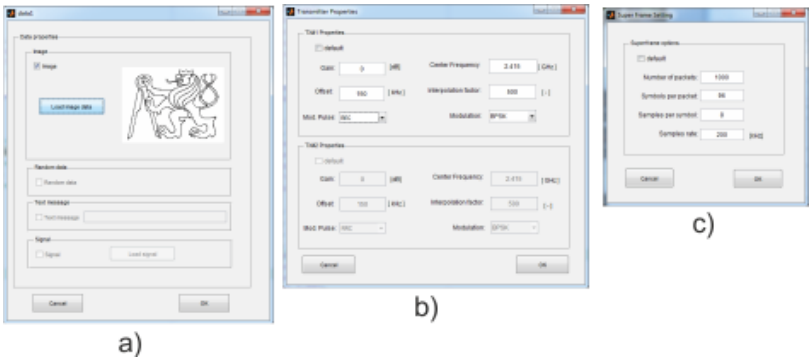

Figure 4.10: Transmitter Data GUI: a), Transmitter properties GUI: b) Superframe properties: c)

4.2.4 Synchronization GUI

This GUI enables to instantly define default pilots for frame and packet synchronization. If the user wants to define his own sequences in order synchronization testing routines, this GUI calls another interface which provide more options. In the initializing function is implemented routine which enables graphical objects used in this GUI only in case that is any transceiver defined in network cloud GUI. This routine is described in following algorithm. Similar initialization is done in the other GUIs as well.

Algoritmus 4.6 Initialization of GUI based on Network Cloud

```
if isappdata(0,'scenario')==0
    set(handles.checkbox1,'Enable','off')
    set(handles.checkbox2,'Enable','off')
    set(handles.pushbutton_packet,'Enable','off')
    set(handles.pushbutton_frame,'Enable','off')
else
    set(handles.checkbox1,'Enable','on')
    set(handles.checkbox2,'Enable','on')
    set(handles.pushbutton_packet,'Enable','on')
    set(handles.pushbutton_frame,'Enable','on')
end
```

This interface is shown on following picture where are is now checked frame synchronization which further define prepared CAZAC sequence as a frame synchronization pilot signal. The packet synchronization is now allowed to modify due to unchecked *checkbox* object.

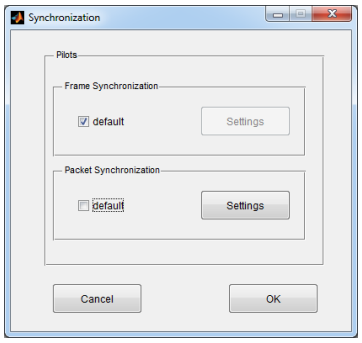

Figure 4.11: Synchronization interface

Frame and packet synchronization interface serves for custom definition of frame or packet synchronization sequences. User can select his own root number and length of this sequences and also number of repeats of this sequences, which causes that pilot signal will have more energy in correlation function therefore pilot signal designed in this way allows to signalize starts of frames/packets even in conditions with lower SNR. In the packet sycnhronization GUI is added option to define Maximum-Length codes as a pilot sequences. These two GUIs are depicted in following figure.

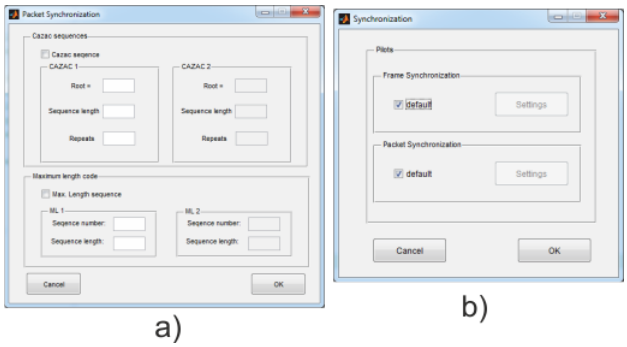

Figure 4.12: Frame synchronization interface: a) Packet synchronization interface: b)

4.2.5 Relay properties GUI

The main purpose of this GUI is to proper configuration of relay which is strictly given by stage schedule. In WPLNC principle relay serves as a receiver for mutiple access stage where it have to process all signals and in next time step relay figures as a broadcaster of these superposed signals. This interface have to handle both variants of relay operational modes. The very important feature of this GUI is a storing and loading signals. Because we are bounded with half-duplex constraint, therefore have to be provided ability to store recieved signal on relay node, reconfigurate the network scenario and load this signal in order to broadcast signal further into network. This routines is depicted in following algorithm.

Algoritmus 4.7 Signal storing routines

```
global savesig
[signalname, pathname] = uiputfile('*.mat',... 'Pick a MATLAB program file');
if signalname==0 && pathname== 0
clear [signalname, pathname]
else
savesig=[pathname, signalname];
end
%loading
[sigName, pathname]=uigetfile('*.mat');
if signalname==0 && pathname== 0
clear [sigName, pathname]
else
loadsig=[pathname, sigName];
end
```

The relay GUI is depicted in following picture, where is relay defined as a receiver for MAC stage there fore is available graphical objects to store received

signal and configure receiver properties. The GUI of receiver properties is vry similar to GUI of transmitter properties execpt of two little changes, therefore is no need to demonstrate this interface again.

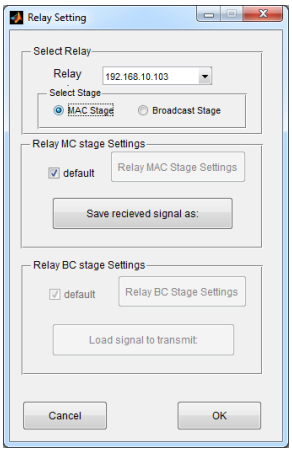

Figure 4.13: Relay interface

4.2.6 Other GUIs

One of remaining GUIs which were not been mentioned yet is Receiver GUI, but the functionality of this GUI is pretty obvious and this GUI does not provide any new or special method to operate. Therefore isn't necessary to mention this GUI in graphical form. The very last GUI and very important one is a GUI which runs the simulations in form of scripts. This GUI have two main options in form of simply simulation and simulation with signal transmission i.e. experimental wireless communication. This GUI is depicted in following figure.

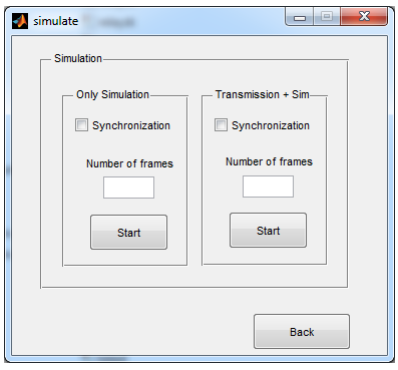

Figure 4.14: Simulaton interface

Simulation GUI also provides definition of number of frames which can be

transmitted and a very useful tool in form of frame synchronization which can be turned of. This desycnhronization is designed in such a way to provide space for sycnhronization algorithm developement and testing. The basic princip of this routine is that frame does not starts with first packet which carry the frame synchronization pilot, but randomly in the data sequences, therefore is needed in this case to transmitts three or more frames. The mistakenly pressed start button is treated by quest dialog window which needs second confirmation.

4.3 Stand-alone application

In previous section we designed GUI for management of the main framework platform. In this section we will expand this GUI into stand-alone application which can be started directly from desktop without using Matlab. To create standalone application is needed to invoke deployment tools which offer the choice of deploy standalone application. Every deployment which provide Matlab with its scripts or programs are realized via *Builders*.

The Matlab Compiler and Builders allow to deploy applications as:

- Stand-alone executables
- C or C++ libraries
- Microsoft.NET or COM components
- Java classes
- Microsoft Excel add-ins

The options of creation C/C++ libraries, Java Classes and others is given by Matlab license management. User have to have purchased for instance Matlab Builder JA to deploy java class from Matlab function. But for our purpose we select deployment of Standalone application.

A deployed application consist of a collection of Matlab-based functions and data packages related with the main process of application which is being deployed. Next step is creation of new *project.prj* and after that we select main program of this framework (main_menu.m) and all related scripts and GUIs which are associated with main file.

After that we can choose if Matlab Compiler Runtime will be contained in the builded project as side file. This allows to portability of this application on devices where is not Matlab installed. Matlab Compiler Runtime is a free compiler wich provide execution of all Matlab commands and functions. The building process duration is given by a number and complexity of involved files in project which is being deployed.

When is builder done with our application it should be indicated with this message:

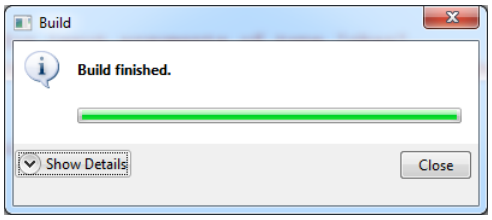

Figure 4.15: Building of stand-alone application

Now we have standalone application ready to use in Matlab home path in folder with name of the *project.prj*. It contain two other subfolders src and distrib. Src serves for computers with installed Matlab copy and the other one is for distribution to other devices. But this will work only if is provided MCR instalation file together with deployed application.

4.3.1 Utilization of framework platform

The utilization of this framework platform is one of the purpose of this whole task. It can serve for demonstration of basic sycnhronization algorithms. It also provide a useful tool how to design signals and instead simple addition of modeled AWGN, user can transmitt this signal in order to further experimental processing. The advantage of this solution of standalone application is that in its basic we created multiple access method to shared USRPs in local network. The laboratory network scheme could be following.

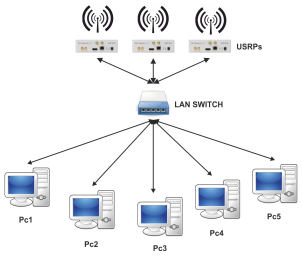

Figure 4.16: Multiple access to USRPs

This solution is much more convenient from a data storage management point of view. Lets consider a fact that multiple access is relized via remote desktop or web based applications. In this cases have to be established rules and privileges in order to allows storing and loading data from computer.

However multiple access to USRPs is very useful tool but it does not guarantee a scheduling of shared resources in form of USRPs. This devices does not provide multitasking. Software radios can execute only one task at a time, therefore have to be scheduling process realized by a human authority which directly controls the multiple access.

Remote control of this framework in form of remote access for instance via web based applications isn't been implemented yet. The main reason of this is safety usage of the expensive USRPs. Moreover in case when error occur it can cause that USRP get stucked and it is necessary to reboot this device. In addition USRPs have separated antennas for Tx/Rx operations, therefore is necessary to proper usage control this antenna inputs manually.

Chapter 5

Results evaluation

Next goal of this thesis is validation of designed platform on real radio experiments. This verification should contain both point-2-point and also cloud oriented experiments in this case 2Tx-Rx and 1Tx-2Rx scenarios. Therefore is this chapter focused on demonstration framework functionality. It this chapter we provide confrontation between designed signal and real signal recieved by USRP. All these results are processed offline in Matlab language.

5.1 2Tx - Rx network scenario

In this section we describe results for this particular scenario which is the first half of the famous butterfly scenario depicted in figure 2.1. Every process based on receiving any kind of signal in practical use should do firstly verification if any signal is received or not. Therefore is a good starting point spectral analysis of received signal in offline signal processing like this. Because in this framework are implemented only linear modulations we can define power spectral density for these modulations.

Power spectrum density of linear digital modulation is then defined as

$$\hat{\mathscr{S}}_S(f) = \frac{1}{Ts}|G\left(f\right)|^2 S_{dq}\left(fTs\right) \tag{5.1}$$

where $G\left(f\right) = \mathscr{F}\left[g(t)\right]$ and S_{dq} is power spectrum density of channel symbols which is gained from

$$S_{dq}\left(F\right) =^d \mathscr{F}_m^F\left[R_q\left[m\right]\right] = \sum_m R_q\left[m\right] e^{-j2\pi Fm}$$

therefore shape of PSD is affected only by symbol pulse and modulation encoder. In Matlab system is provided function for PSD evaluation called *pwelch*. Usualy is this function called as

$$[Pxx, f]\text{=pwelch}(x, nwin, noeverlap, nfft, fs)$$

where x is a vector with the time series (in our case received signal), *nwin* is an integer (then a Hamming window of that length is used), *noverlap* is an integer (a fraction of the FFT length) that indicates the desired overlap, *nfft* is and integer giving the length N of the FFT (should corresponds with window length *nwin*) and finally sampling frequency *fs*. The results of this function are *Pxx*, a vector with the PSD and *f*, a vector with the corresponding frequencies.

In this scenario is received signal created by superposition of two signals from *Tx1* and *Tx2*. Next figure shows PSD of designed signal in comparison with PSD of received superposed signal

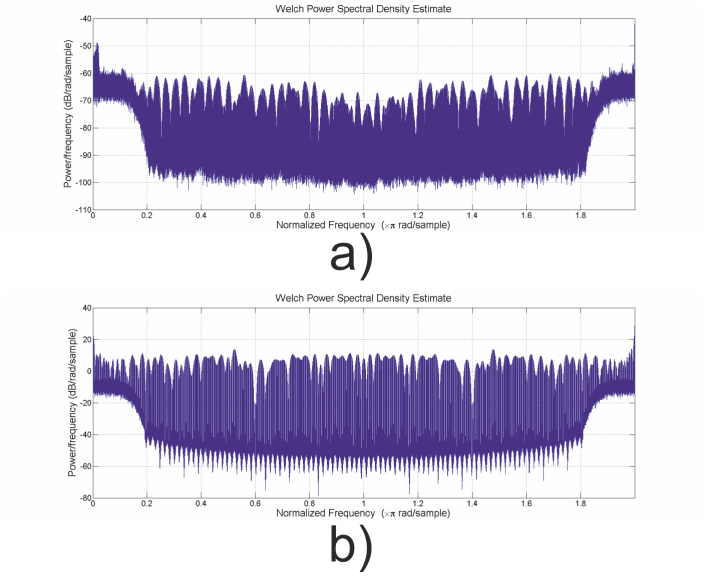

Figure 5.1: a) PSD of recieved signal, b) PSD of designed signal

Now we know that recieved signal contains transmitted signal. In order to prove that received signal contain any useful data we apply correlation function to obtain packet synchronization which will be mainly demonstrated on this example. This signal was designed with CAZAC sequences signalising starts of packets. This signalisation is depicted in figure 5.2. The CAZAC sequences for this example have roots $r_1 = 31$, $r_2 = 7$ and common length $N = 113$.

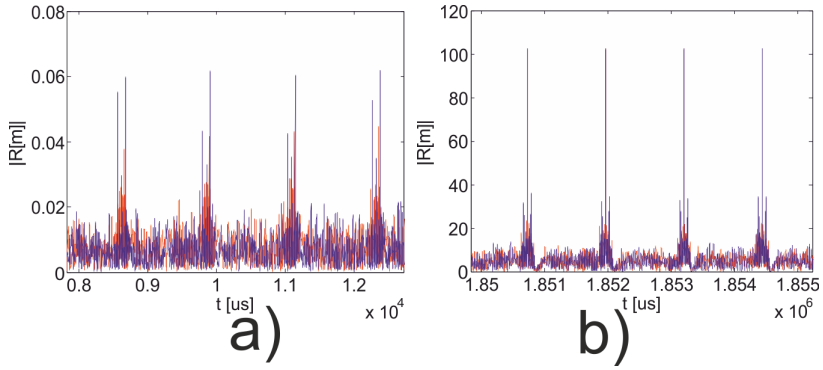

Figure 5.2: Packet synchronization with CAZAC sequences: a) recieved signal, b) signal model

This picture shows that signal model is perfectly synchronized and every peaks have same amplitude. On the other hand received signal is slightly shifted due to time difference between *Tx1* and *Tx2*. In the received signal are also very high side peaks of packet synchronization which can caused complications in further signal processing.

Next example is the same scenario but instead of cazac sequences for packet synchronization are used Maximum Length codes. These codes have a length $N = 125$ and are orthogonal to each other.

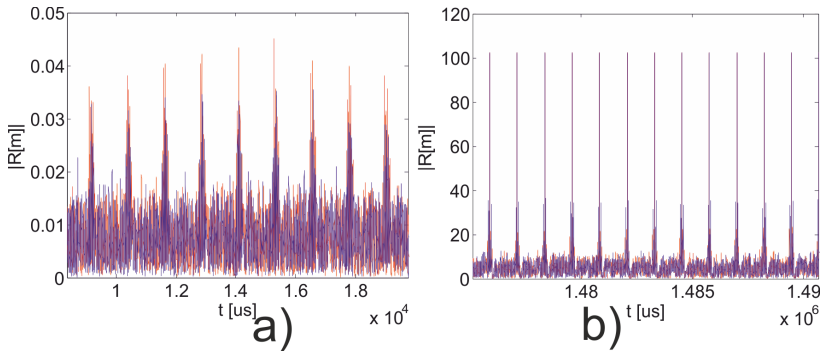

Figure 5.3: Packet synchronization with ML codes: a) received signal, b) designed signal

In this picture is very well demonstrated stretched maximum of all synchronization peaks in received signal therefore better results for synchronization provides CAZAC sequences. Also is very important to know that all these results are with usage of RRC modulation pulse. The modulation pulse has influence on the shape of PSD as it was mentioned on the beginning of this section.

5.2 1Tx -2Rx network scenario

In order to completion of butterfly scenario, in this section are demonstrated results of the second half of butterfly scenario. The reason why we have to separate this processes into two wholy independent operations is half-duplex constraint already mentioned in chapter 3. Due to this constraint we are forced to slice more complex network scenarios on these triangles to be able emulate advanced network scenarios. But this framework is not designed for emulating of advanced network topologies.

The following figure shows PSDs of recieved signals in both destination points.

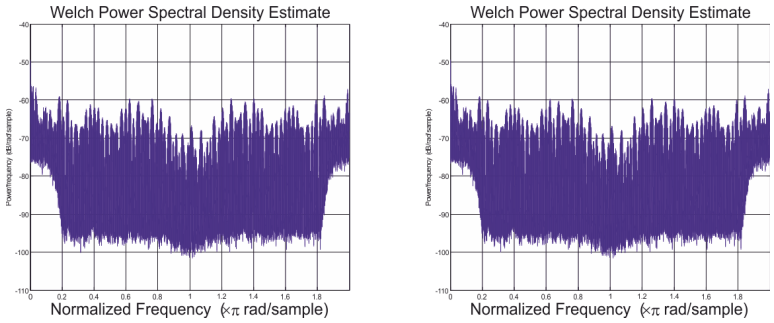

Figure 5.4: PSD of 1Tx-2Rx scenario

This picture shows that recieved signals are very similar to each other, but this is due to network scenario. It can be seen that the received signals are with lower power than in scenario 2Tx-Rx. This feature can cause a problems with data decoding and maybe with proper synchronization as well. In the next figure is depicted packet synchronization of this scenario. It can be seen that packet synchronization have not so sharp peaks like in the previous scenario.

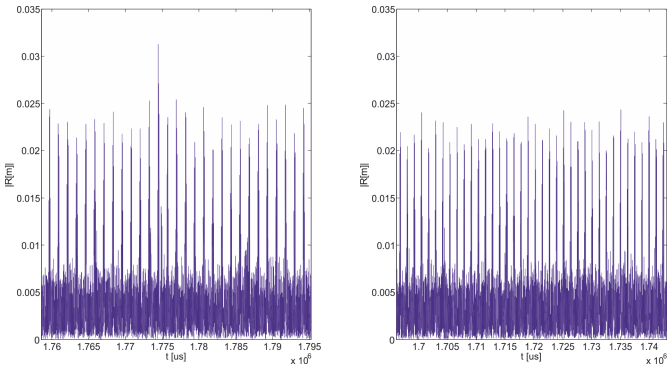

Figure 5.5: Packet sycnhronization in 1Tx-2Rx scenario

5.3 Tx - Rx network scenario

This scenario is elementary in common wireless systems but with one difference. In the regular point-2-point wireless networks are generaly used directional antennas, but in this case are USRPs equipped with omnidirectional antennas. It should not affect this demonstration due to very short distances of signal transmission. In this demonstration is used REC modulation pulse which is provided option of this framework. The second change is the fact the transmitted signal will be purposely desynchronized from the frame synchronization point of view. Which is also provided option of this platform. The following figure show PSD of REC pulse in this scenario.

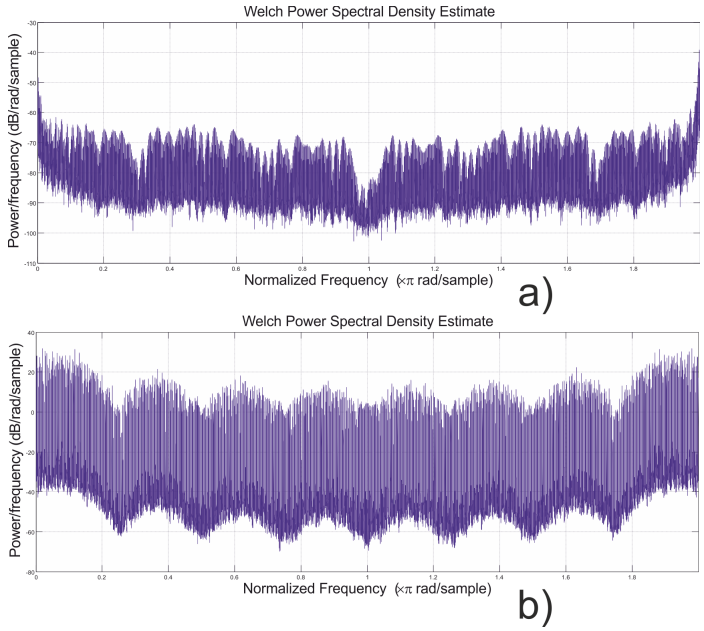

Figure 5.6: PSD with REC pulse in Tx-Rx scenario: a) received signal, b) designed signal

This figure shows that better spectral properties for wireless communication have a RRC pulse. But in some cases REC pulse can provide properties sufficient enough to implementation for instance in algorithm developement. for sycnhronization. The following picture shows frame and packet synchronization where is interesting first picture which shows two starts of frame. The transmission was processed in such a way that beginning was not with the very first packet which carry the pilot signal indicating frame synchronization. Due to this feature this platform can be used for instance for synchronization algorithm developement.

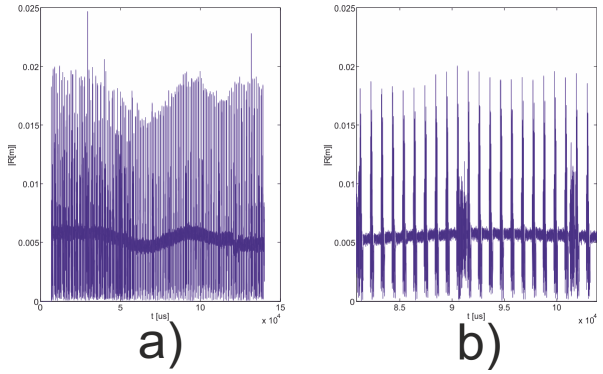

Figure 5.7: Tx-Rx scenario: a) frame synchronization, b) packet synchronization

Chapter 6

Conclusion

In this thesis was designed framework platform which define a basic interface protocol in order to common rules establishment for proper communication in wireless network. It provides functionality in multiple network scenarios for experimental wireless transmission. Framework also provide selection between two implemented modulations and two modulation pulses. It was designed user-friendly Graphical User Interface in order to provide comfortable solution of framework controling. This framework was also extended into standalone application which provides portability of this testbed to any other devices which also provide multiple access to shared Universal Software Radio Peripherals. But unfortunatelly the main purpose of this framework i.e to serve for demonstration of basic WPLNC techniques such NCM encoder is not properly provided. Despite of high amount of effort was spent in order to guaranty a perfect frame synchronization and symbol timing necessary for correct HDF relaying strategy, I was unable to achieve these properties. The cause of this unability to achieve sufficient synchronization properties is an non-optimal choice of based developement environment for this platform. This choice I made in virtue of my philosophy of obstacle solving processes. The key feature of this philosophy is that before any investment in device which will lead to solution, is firstly necessary to investigate all possible ways how to achieve the desirable result without necessity of investment. If there is no other way how to accomplish the task, then is a right moment to investment.

6.1 Future work

In order to fulfil desired synchronization of Tx/Rx processes which also warrants the perfect frame sycnhronization and symbol timing, framework should be implemented into GNU radio software developement toolkit which allows usage of external clock source to provide precise process sycnhronization directly on physical layer as it requested in WPLNC fundamentals. Unfortunatelly this solution is conditioned on investment in to an expensive synchronization device.

Framework designed in this way should be able to provide proper WPLNC techniques namely HDF decoding processes.

The next goal for future work should be remote access not only via ssh and other remote desktop solutions but via web based programs written in Java based languages. This solution of remote access will be very sophisticated. But complexity of this task is enourmous and if we consider the fact that the results from sophisticated solution and easy realizable solution will be the same, nobody will hesitate.

Bibliography

1. R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," Information Theory, IEEE Transactions on, vol. 46, no. 4, pp. 1204 –1216, Jul. 2000.

2. S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," Information Theory, IEEE Trans- actions on, vol. 49, no. 2, pp. 371–381, 2003.

3. R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," Information Theory, IEEE Transactions on, vol. 51, no. 8, pp. 2745–2759, 2005.

4. S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "Xors in the air: Practical wireless network coding," Networking, IEEE/ACM Transactions on, vol. 16, no. 3, pp. 497–510, 2008.

5. S. Zhang, S. C. Liew, and P. P. Lam, "Hot topic: physical-layer network coding," in Proceed- ings of the 12th annual international conference on Mobile computing and networking, ser. MobiCom '06. New York, NY, USA: ACM, 2006, pp. 358-365

6. B. Rankov and A. Wittneben, "Achievable rate regions for the two-way relay channel," in Information Theory, 2006 IEEE International Symposium on, 2006, pp. 1668–1672.

7. P. Popovski and H. Yomo, "Physical network coding in two-way wireless relay channels," in Communications, 2007. ICC '07. IEEE International Conference on, 2007, pp. 707–712.

8. B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference through structured codes," Information Theory, IEEE Transactions on, vol. 57, no. 10, pp. 6463 –6486, Oct. 2011.

9. R. Zamir, "Lattices are everywhere," in Information Theory and Applications Workshop, 2009, 2009, pp. 392–421.

10. J. Sykora and A. Burr, "Network coded modulation with partial side-information and hierarchical decode and forward relay sharing in multi-source wireless network," in Wireless Conference (EW), 2010 European, 2010, pp. 639–645.

11. Sykora, J.; Burr, A., "Layered Design of Hierarchical Exclusive Codebook and Its Capacity Regions for HDF Strategy in Parametric Wireless 2-WRC," Vehicular Technology, IEEE Transactions on , vol.60, no.7, pp.3241,3252, Sept. 2011

12. Sykora, J.; Jorswieck, E.A., "Network Coded Modulation with HDF Strategy and Optimized Beam-Forming in 2-Source 2-Relay Network," Vehicular Technology Conference (VTC Fall), 2011 IEEE , vol., no., pp.1,6, 5-8 Sept. 2011

13. Hynek, T.; Sykora, J., "Hierarchical Decode & Forward strategy in IR-UWB communication systems," Future Network & Mobile Summit (FutureNetw), 2011 , vol., no., pp.1,8, 15-17 June 2011

14. Cisco Visual Networking Index: Forecast and Methodology, 2012–2017

15. DIWINE - Dense cooperative wireless cloud network - http://www.diwine-project.eu