ICT- 619555 RESCUE

Deliverable D2.3 Version

Feasibility Assessment in Model Based Environments

Contractual Date of Delivery to the CEC:	2015-10-31
Actual Date of Delivery to the CEC:	2015-11-06
Editor:	Jacek Wszołek
Author(s):	Sebastian Sośnik, Valtteri Tervo, Jacek Wszołek, Xin He, Shen QIAN, Jiancao Hou, Marek Sikora, Krzysztof Łoziak, Janusz Gozdecki, Maximilian Matthe, Jarosław Bułat, Christian Schneider, Łukasz Trzeciakowski, Szymon Szott, Hicham Khalife, Andrzej Pach
Participant(s):	*AGH, FQS, JAIST, UOULU, TUD, TUIL, UNIS, TCS*
Work package:	*WP2 - Code and algorithm design and optimization*
Estimated person months:	**47**
Security:	**PU**
Nature:	**R**
Version:	1.0
Total number of pages:	67

Abstract: The RESCUE project proposes a novel multi-route communication technology design targeted for multi-hop networks, based on cooperative decode-and-forward (DF) relaying allowing intra-link errors and utilizing distributed turbo codes. This deliverable provides a report of the RESCUE Lossy Forwarding (LF) concept implementation and evaluation in a software defined radio platform. Performance analysis proved that RESCUE LF technique performs always better than the baseline Selective Decoding-and-Forward (SDF) with or without joint decoder in the destination node. In the case when the relay node is close enough to the source node to establish a lossless radio link the performance of the RESCUE LF is close to the SDF with joint decoder. However, when the distance between source and relay node is too large to establish a lossless radio link, and there are errors in the relay node after decoding, the RESCUE LF significantly outperforms SDF.

Keyword list: GNU Radio, Software Defined Radio, distributed turbo code, joint decoder, lossy forwarding, multi-hop communication, wireless networks,

Disclaimer:

Executive Summary

The main goal of the RESCUE project – "Links-on-the-fly Technology for Robust, Efficient, and Smart Communication in Unpredictable Environments" is to enable wireless communication in devastated areas where the partially destroyed communication infrastructure can operate only with the limited range. In case of natural disasters like earthquakes, tsunami or fire over a large area, one of the most important tasks is to extend the wireless network coverage in order to enable fast and efficient rescue operations. Therefore there is a need for robust technologies that can allow coverage extension in a reliable manner. One of the possible and promising solutions to this problem is the Lossy Forwarding concept we propose in RESCUE.

Lossy Forwarding is a distributed source coding (DSC) technology. In the conventional Decode-and-forward relaying, relay forwards only correctly decoded frames. Lossy forwarding allows a relay to forward a message even if errors have been detected after decoding. At the destination, a joint decoding technique exploits the high correlation of the messages received via different network paths thus extending wireless network range.

Deliverable D2.3 is a report of the implementation and verification of the RESCUE concept in the Software Defined Radio (SDR) platform based on USRP hardware and GNU radio framework. This implementation constitutes a step towards the hardware implementation of the RESCUE concept developed within WP1, WP2 and WP3 in a full radio proof of concept. The research and initial ideas have thus far been verified with MATLAB simulations. However, critical parts of the code, especially requiring a lot the computational power, were also implemented in C/C++.

This deliverable consists of two chapters. In section 2, a detailed description of implemented software modules and GNU Radio blocks was presented. This section is divided into three main subsections. In subsection 2.2 a RESCUE source node implementation is presented. Subsection 2.3 consists of description of the RESCUE destination node and finally subsection 2.4 presents the relay node.

Section 3 contains simulation results analysis obtained from the use of software defined radio implementation described in section 2. It starts with subsection 3.1 where the simulation software and models are presented. Then, in subsections 3.2 and 3.3 simulation scenarios, parameters and used channel models are described. Subsections 3.4 and 3.5 provide a detailed simulation analysis of the Lossy Forwarding concept in various scenarios and use cases. Finally, subsection 3.6 summarizes the simulation analysis results.

Performance analysis of the RESCUE Lossy Forwarding concept developed within WP1 and WP2 in the software defined radio platform allows to draw the following conclusions. First of all, the RESCUE Lossy Forwarding technique performs always better than the baseline SDF with or without joint decoder in the destination node.

In the case when the relay node is close enough to the source node to establish a lossless radio link (location A) the performance of the RESCUE LF is close to the SDF with (RESCUE) joint decoder. Lossless radio link between source and relay implicates no errors at the relay node, so in that case there is no lossy forwarding. However, an important conclusion is that RESCUE coding algorithms do not introduce loss in non-LF scenarios.

On the contrary, when the distance between source and relay node is too far to establish a lossless radio link and there are errors in the relay node after decoding, the performance of the RESCUE LF is significantly better than SDF. In that case, RESCUE can be viewed as a form of estimate-and-forward relaying strategy, which is a suboptimal version of the optimal compress-and-forward strategy to be applied when successful decoding is not possible at the relay. This leads us to conclude that *i)* RESCUE LF performs near optimally through the whole range of relay positions and *ii)* RESCUE coding structure can improve performance of the SDF.

Interestingly, Software defined radio platform performance analysis have shown that the gain is biggest when joint decoder in the destination node decodes jointly two copies of the transmitted frame. Sending the third and fourth copy to the destination usually does not add significant gain. This is due to the errors in the frame header, which in this case must be discarded.

In summary, the most important D2.3 contributions are the following:

- implementation of the RESCUE environment, which can be used by the whole GNU Radio community,
- presentation of the RESCUE LF concept simulation results and its performance comparison with SDF joint decoding strategy in many scenarios (QPSK, DQPSK, 16QAM, D16QAM, with and without waveform, additive white Gaussian noise (AWGN) and fading channel)
- feedback from practical verification to theoretical research.

Authors

Partner	Name	Phone / Fax / e-mail
AGH University of Science And Technology	Jacek Wszołek	jacek.wszolek@agh.edu.pl
	Marek Sikora	sikora@kt.agh.edu.pl
	Jarosław Bułat	jaroslaw.bulat@agh.edu.pl
	Krzysztof Łoziak	krzysztof.loziak@agh.edu.pl
	Janusz Gozdecki	gozdecki@agh.edu.pl
	Szymon Szott	szott@kt.agh.edu.pl
	Andrzej Pach	pach@kt.agh.edu.pl
FQS Poland	Sebastian Sośnik	s.sosnik@fqs.pl
	Łukasz Trzeciakowski	l.trzeciakowski@fqs.pl
Japan Advanced Institute of Science and Technology (JAIST)	Xin He	hexin@jaist.ac.jp
	Shen QIAN	shen.qian@jaist.ac.jp
University of Oulu (UOULU)	Valtteri Tervo	valtteri.tervo@ee.oulu.fi
Technical University Dresden (TUD)	Maximilian Matthe	maximilian.matthe@ifn.et.tu-dresden.de
University of Surrey (UNIS)	Jiancao Hou	j.hou@surrey.ac.uk
Technical University Ilmenau (TUIL)	Christian Schneider	christian.schneider@tu-ilmenau.de
Thales Communications & Security (TCS)	Hicham Khalife	hicham.khalife@thalesgroup.com

Table of Contents

List of Acronyms and Abbreviations

Term	**Description**
ACC	Accumulator
ADC	Analog to Digital Converter
AWGN	Additive White Gaussian Noise
BAS	Baseline
BER	Bit Error Rate
BSM	Basics Simulation Model
CC	Convolutional Code
CRC	Cyclic Redundancy Check
DAC	Digital to Analog Converter
DF	Decode-and-forward
ESM	Extended Simulation Model
EXIT	EXtrinsic Information Transfer
FER	Frame error rate
GI	Global Iteration
GMSK	Gaussian Minimum Shift Keying
GR	GNU Radio
GRC	GNU Radio Companion
HER	Header error rate
IL	Interleaver index
LF	Lossy Forwarding
LI	Local Iteration
LLR	Log-Likelihood Ratio
M2M	Machine to machine
MCS	Modulation and Coding Scheme
MI	Mutual Information
MIC	Message Integrity Check
MSP	Modified Set Partitioning
OFDM	Orthogonal Frequency Division Multiplex
OTA	Over-the-air
PC	Personal Computer
PDU	Protocol Data Unit
PHY	Physical Layer
POC	Proof of Concept
PPDU	Physical Layer PDU
PSK	Phase-shift keying
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RCA	RESCUE Coding Algorithms
SDF	Selective decode-and-forward
SNR	Signal-to-Noise Ratio
TS	Toy Scenario
USRP	Universal Software Radio Peripheral
V2V	Vehicle to vehicle

Port Colors

This document contains several GNU Radio Companion (GRC) flow graphs. In those graphs the block interfaces (sockets or ports) are coloured to indicate their data type. Connections between block are possible only, when the connected sockets are of the same type. Most commonly used port color-type mapping is presented in the Table 0-1.

Table 0-1 Port/socket colour mapping in GNU Radio Companion

Port Color	**Type**
Blue	Complex Float 32
Orange	Float 32
Pink	Integer 8
Light Grey	Async Message

1. Introduction

The RESCUE project – “Links-on-the-fly Technology for Robust, Efficient, and Smart Communication in Unpredictable Environments” – aims to enable wireless communication in devastated areas where the partially destroyed communication infrastructure can operate only with the limited range. Moreover, RESCUE investigates the use of Lossy Forwarding in V2V scenarios where losses are caused by vehicles mobility and possible obstacles (non-Line of sight communications).

Lossy Forwarding is an innovative DSC technology. Unlike in the conventional Decode-and-forward relaying, with lossy forwarding a relay forwards a message regardless whether errors have been detected after decoding. At the destination, a joint decoding technique exploits the high correlation of the messages received via different network paths. By allowing the relay nodes to forward erroneous messages, the Lossy forwarding technique enables extending wireless network range.

Deliverable D2.3 reports the implementation and verification of the RESCUE concept in the Software Defined Radio (SDR) platform based on USRP devices and GNU Radio frameworks. This implementation constitutes a step towards the hardware implementation of the RESCUE concept developed within WP1, WP2 and WP3 in a full radio proof of concept. The research and initial ideas have thus far been verified with MATLAB simulations. However, critical parts of the code, especially requiring a lot the computational power, were also implemented in C/C++.

At this stage of the verification of the RESCUE concept, first the core ideas were implemented in a C++-based simulator which we refer to as the RCA (Rescue Coding Algorithms). The RCA simulator consists of 37 files with over 2700 lines of C++ code developed by the project team and over 81 000 lines of third party libraries. RCA assumes a realistic channel model, however, it omits certain implementation-specific aspects including receiver synchronisation and the phase ambiguity problem. Additionally, problems related to imperfect header decoding were not considered in RCA.

Subsequently, the C++ code integrating the RESCUE coding and decoding algorithms was implemented in the GNU Radio software platform. The GNU Radio implementation uses real-world synchronisation algorithms. It consist of 26 blocks developed by the project consortium, grouped in 244 files containing over 10 800 lines of source code.

Not every part of our research have been integrated in GNU Radio. Examples of implemented modules but not integrated with the software defined radio platform are the Signal to Noise Ratio (SNR) estimator and the Log-Likelihood-Ratio (LLR) calculator which utilizes the feedback from the decoder to enhance the accuracy of LLRs estimation. They were not integrated as a separate block yet because GNU Radio does not support feedback loop in the flow graph. In that case both SNR estimator and LLR calculator must become a part an existing GNU Radio blocks.

Finally, deliverable D2.3 consists of two main parts. In Section 2, we provide a detailed description of the implementation the RESCUE Lossy Forwarding concept as realized in the SDR platform (i.e., GNU Radio). In Section 3, we provide the results of the simulation analysis performed with the use of the previously developed implementation.

2. Implementing the RESCUE Concept in GNU Radio

A software defined radio (SDR) is a radio communication system in which the number of hardware modules are limited to the necessary minimum. Most of the system modules are implemented in software for both the TX (transmit) and RX (receive) paths. Therefore, we can divide the SDR radio communication system into the hardware and software part.

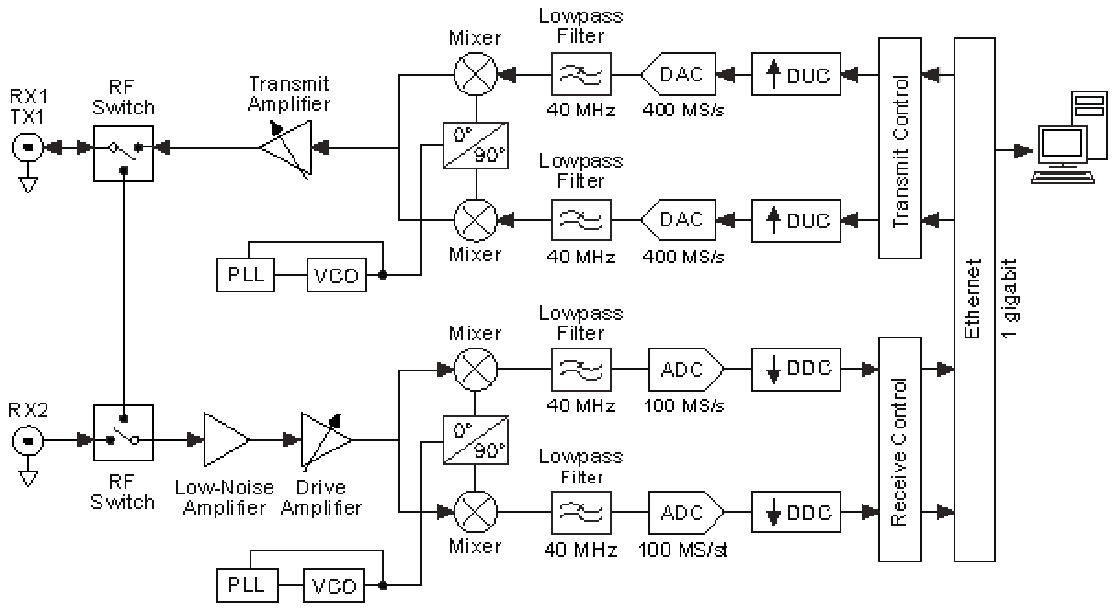


Figure 2-1. Architecture of the USRP N210 device (source: Ettus Research)

The hardware part consists of: transmitting and receiving antenna(s), amplifiers (both the TX amplifier and the low-noise RX amplifier), mixers for analog up and down conversion as well as low pass filters for filtering out-of-band signals. The Universal Software Radio Peripheral – USRP N210 (presented in Figure 2-1) which is being used in RESCUE has a 40 MHz low pass filter. Therefore it supports up to 40 MHz analog bandwidth. The RX path consist of: low pass filter, analog to digital converter (ADC), including sampler and quantizer. The TX path includes a digital to analog converter (DAC) which precedes the low-pass filter. The ADC and DAC are connected through digital down/up converters to transmit/receive controls which provide an interface to the host processor. The USRP N210 can be connected to a PC using Gigabit Ethernet and UDP for the delivery of digital streams.

The host PC requires the USRP hardware driver software to correctly receive and/or transmit the stream to/from the USRP and dedicated application software to process the stream. The USRP N210 is officially supported by the following applications:

- GNU Radio,
- LabVIEW,
- MATLAB/Simulink.

MATLAB [MWHP] and LabVIEW [NIHP] are commercial products. GNU Radio is a free, open source development toolkit for programming SDRs. It is widely used in both academic and commercial environments to support wireless communications research and also to implement commercial radio systems. GNU Radio uses C++ for developing signal processing blocks and Python for writing scripts which build and control the GNU Radio flow-graphs. GNU Radio is a fast and efficient environment, and therefore perfectly suited for the RESCUE project as an application to process the software radio signals.

The following section describes the implementation of the RESCUE concept in GNU Radio. It starts with Section 2.1 which presents a methodology of establishing the over the air radio transmission between two USRP devices. The section 2.2 provides information about the implementation of the RESCUE source node. The destination node implementation is described in Section 2.3 and, finally, Section 2.4 provides a description of the RESCUE Relay Node implementation.

2.1 Radio communication over SDR

The main goal of RESCUE is to implement a radio communication solution for lossy forwarding concept. One of the most important parts of the project realization is to develop a working setup, as a proof-of-concept. Thus, a dedicated testbed has been set up in order to perform experiments with real-time radio transmissions over the software-defined radio units (the USRP N210). For performance reasons it is important to have the USRPs directly connected to laptop computers with the GNU Radio environment running. It not recommended to interconnect USRPs and laptops through an L2 switch. Otherwise performance and instability issues occurs, which can cause the software to stop responding.

The testbed topology for achieving communication over a radio channel is presented in the Figure 2-2. One of the USRPs is used as a transmitter, while the other – as the receiver. The testbed is configured to operate on the unlicensed 434 MHz frequency. Each Ethernet connection between the USRP and its corresponding laptop is organized as a separate IP subnet.

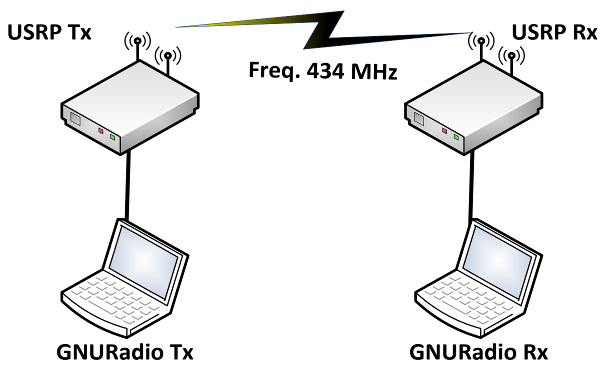
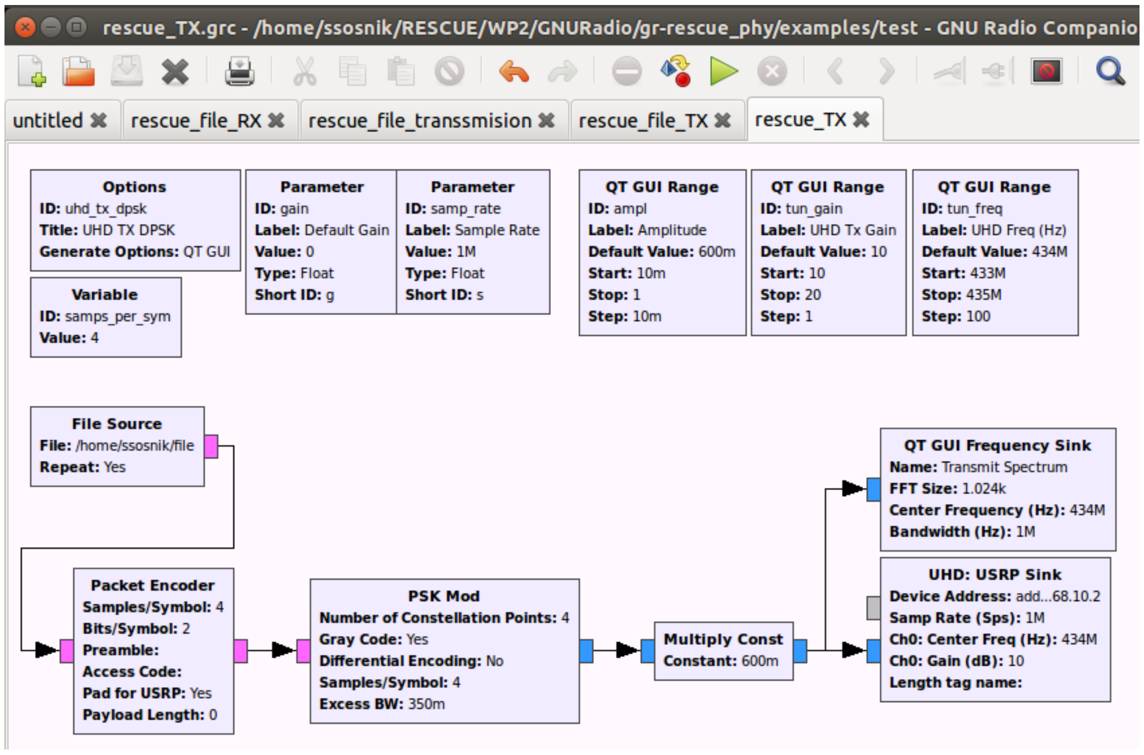


Figure 2-2 Testbed topology

In order to perform the basic transmission test using the USRPs with PSK modulation, we have prepared the transmitter and the receiver configurations, the block diagrams of which are presented in Figure 2-3 and Figure 2-4, respectively. The diagrams were prepared using the GNU Radio software development toolkit. Communication between the transmitter and the receiver was performed by radio waves without any external synchronization.



rescue_TX.grc - /home/ssosnik/RESCUE/WP2/GNURadio/gr-rescue_phy/examples/test - GNU Radio Companio

Options
ID: uhd_tx_dpsk
Title: UHD TX DPSK
Generate Options: QT GUI

Variable
ID: samps_per_sym
Value: 4

Parameter
ID: gain
Label: Default Gain
Value: 0
Type: Float
Short ID: g

Parameter
ID: samp_rate
Label: Sample Rate
Value: 1M
Type: Float
Short ID: s

QT GUI Range
ID: ampl
Label: Amplitude
Default Value: 600m
Start: 10m
Stop: 1
Step: 10m

QT GUI Range
ID: tun_gain
Label: UHD Tx Gain
Default Value: 10
Start: 10
Stop: 20
Step: 1

QT GUI Range
ID: tun_freq
Label: UHD Freq (Hz)
Default Value: 434M
Start: 433M
Stop: 435M
Step: 100

File Source
File: /home/ssosnik/file
Repeat: Yes

Packet Encoder
Samples/Symbol: 4
Bits/Symbol: 2
Preamble:
Access Code:
Pad for USRP: Yes
Payload Length: 0

PSK Mod
Number of Constellation Points: 4
Gray Code: Yes
Differential Encoding: No
Samples/Symbol: 4
Excess BW: 350m

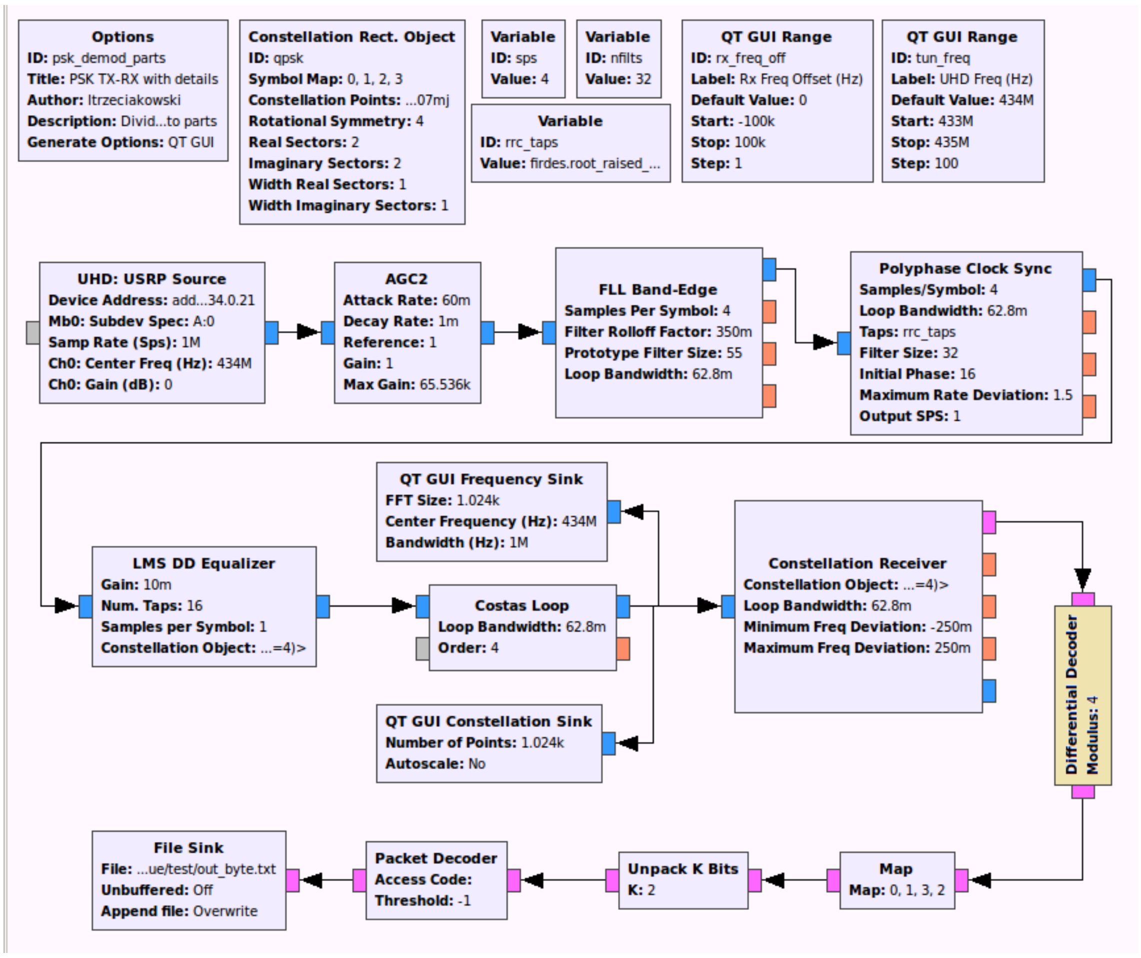
Multiply Const
Constant: 600m

QT GUI Frequency Sink
Name: Transmit Spectrum
FFT Size: 1.024k
Center Frequency (Hz): 434M
Bandwidth (Hz): 1M

UHD: USRP Sink
Device Address: add...68.10.2
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): 434M
Ch0: Gain (dB): 10
Length tag name:

Figure 2-3 Transmitter block diagram in GNU Radio

The transmitter block diagram presented in Figure 2-3 is based on a GNU Radio example file (`uhd_th_dpsk.grc`), in which the DPSK modulation was replaced with PSK. The transmitter application is presented in Figure 2-5. This application consists of four objects: three sliders allow to change the parameters of the transmitted signal and a spectrum analyzer which shows the spectrum of the transmitted signal.



Options
ID: psk_demod_parts
Title: PSK TX-RX with details
Author: ltrzeciakowski
Description: Divid...to parts
Generate Options: QT GUI

Constellation Rect. Object
ID: qpsk
Symbol Map: 0, 1, 2, 3
Constellation Points: ...07mj
Rotational Symmetry: 4
Real Sectors: 2
Imaginary Sectors: 2
Width Real Sectors: 1
Width Imaginary Sectors: 1

Variable
ID: sps
Value: 4

Variable
ID: nfilts
Value: 32

Variable
ID: rrc_taps
Value: firdes.root_raised_...

QT GUI Range
ID: rx_freq_off
Label: Rx Freq Offset (Hz)
Default Value: 0
Start: -100k
Stop: 100k
Step: 1

QT GUI Range
ID: tun_freq
Label: UHD Freq (Hz)
Default Value: 434M
Start: 433M
Stop: 435M
Step: 100

UHD: USRP Source
Device Address: add...34.0.21
Mb0: Subdev Spec: A:0
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): 434M
Ch0: Gain (dB): 0

AGC2
Attack Rate: 60m
Decay Rate: 1m
Reference: 1
Gain: 1
Max Gain: 65.536k

FLL Band-Edge
Samples Per Symbol: 4
Filter Rolloff Factor: 350m
Prototype Filter Size: 55
Loop Bandwidth: 62.8m

Polyphase Clock Sync
Samples/Symbol: 4
Loop Bandwidth: 62.8m
Taps: rrc_taps
Filter Size: 32
Initial Phase: 16
Maximum Rate Deviation: 1.5
Output SPS: 1

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 434M
Bandwidth (Hz): 1M

LMS DD Equalizer
Gain: 10m
Num. Taps: 16
Samples per Symbol: 1
Constellation Object: ...=4)>

Costas Loop
Loop Bandwidth: 62.8m
Order: 4

Constellation Receiver
Constellation Object: ...=4)>
Loop Bandwidth: 62.8m
Minimum Freq Deviation: -250m
Maximum Freq Deviation: 250m

Differential Decoder
Modulus: 4

QT GUI Constellation Sink
Number of Points: 1.024k
Autoscale: No

File Sink
File: ...ue/test/out_byte.txt
Unbuffered: Off
Append file: Overwrite

Packet Decoder
Access Code:
Threshold: -1

Unpack K Bits
K: 2

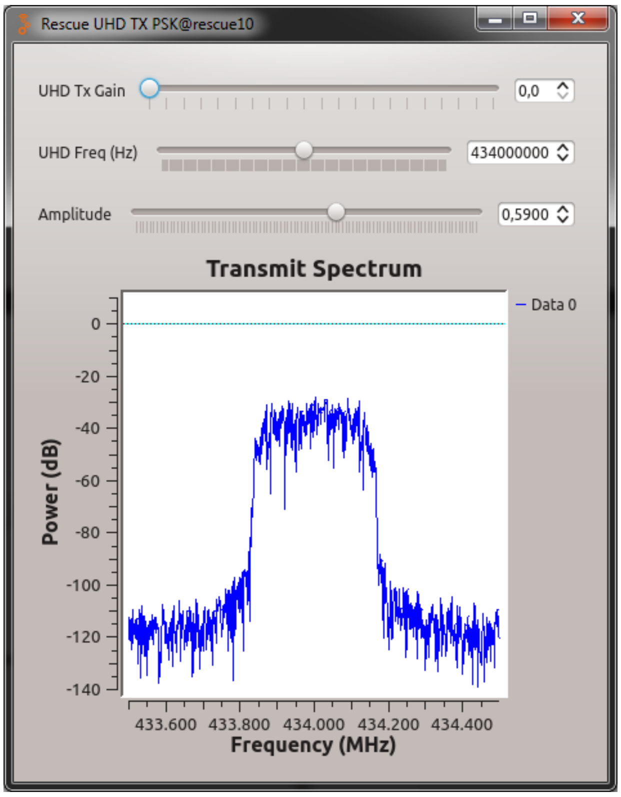
Map
Map: 0, 1, 3, 2

Figure 2-4 Receiver block diagram in GNU Radio

The receiver presented in Figure 2-4 consists of the following main blocks:

- UHD USPR Source – the USRP source block which receives samples and writes to a stream;
- AGC – a high performance Automatic Gain Control class with attack and decay rates;
- Polyphase Clock Sync – performs the time synchronization procedure for PAM signals by minimizing the derivative of the filtered signal, which in turn maximizes the SNR and minimizes ISI;
- LMS DD Equalizer – implements an LMS-based decision-directed equalizer. It uses a set of weights *w* to correlate with the inputs *u* and a decision is then made from this output. The detection error is used to update the weight vector;
- Costas Loop – locks to the center frequency of a signal and down-converts it to baseband;
- Constellation Receiver – makes hard decisions about the received symbols (using a constellation object) and also fine tunes phase synchronization;
- Map – maps an incoming signal to the value in the map;
- Unpack K Bits –converts a byte with *k* relevant bits to *k* output bytes with 1 bit in the LSB;
- Packet Decoder;
- Visualization QT modules (Frequency Sink, Constellation Sink and Time Sink);
- Several modules responsible for changing decoder parameters in real time using sliders.

- The Differential Decoder block is marked yellow because it is bypassed in this scenario to show that the correct transmission between USRP Tx and USRP Rx can be performed both with and without differential coding. Both of them were correct. Figure 2-6 presents the running decoder application.


Rescue UHD TX PSK@rescue10

UHD Tx Gain 0,0

UHD Freq (Hz) 434000000

Amplitude 0,5900

Figure 2-5 UHD transmitter application used for basic radio connectivity tests

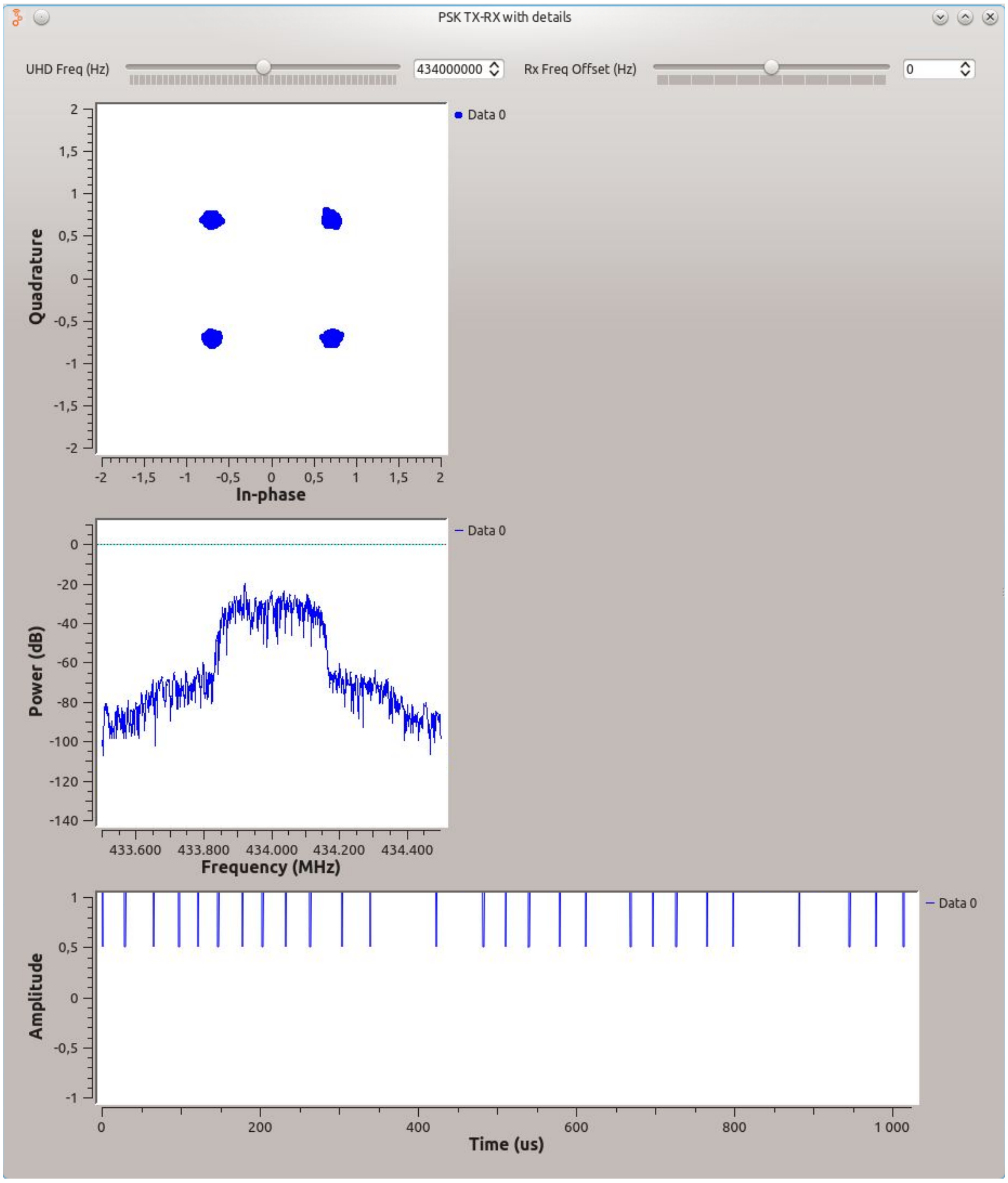
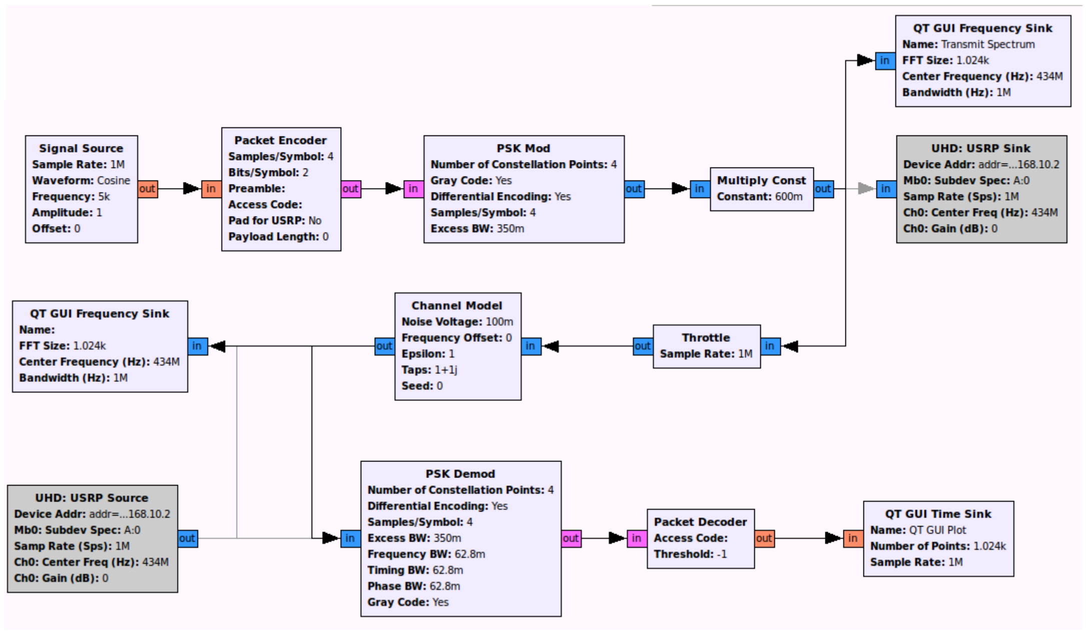


Figure 2-6 UHD receiving application used for tests

The final verification of the successful radio transmission was done by a comparison of the input message source from USRP TX and a reference file, which was stored locally at the RX USRP side. Due to high SNR values the Packet Decoder block discovered 100% correctly received and decoded symbols. However, the TX application allows to perform some parameter degradations: decreasing transmitter gain or signal amplitude which can result with synchronisation lost and lack of packets decoding.

In the following: we proceed with the controlled environment of the SDR software platform by replacing the radio channel with a channel model in GNU Radio. The controlled environment allows us to have the whole environment under control and working on one machine. Fortunately, the modularity of GNU Radio allows reverting to the real radio channel by means of a simple procedure (i.e., diverting the stream flow from the channel model to/from the USRP Sink/Source blocks responsible for the TX/RX paths in the devices themselves).

As the next step we performed a PSK modulation test. Its GNU Radio flow graph is shown in Figure 2-7. For this test, float numbers forming a cosine plot were generated with the sample rate of 1M. Then the numbers were converted to bytes and modulated using PSK. On the RX side the numbers were demodulated, converted to floats, and plotted. When the plot forms a cosine function we assume that the demodulation was successful.



Signal Source — Sample Rate: 1M; Waveform: Cosine; Frequency: 5k; Amplitude: 1; Offset: 0

Packet Encoder — Samples/Symbol: 4; Bits/Symbol: 2; Preamble: ; Access Code: ; Pad for USRP: No; Payload Length: 0

PSK Mod — Number of Constellation Points: 4; Gray Code: Yes; Differential Encoding: Yes; Samples/Symbol: 4; Excess BW: 350m

Multiply Const — Constant: 600m

QT GUI Frequency Sink — Name: Transmit Spectrum; FFT Size: 1.024k; Center Frequency (Hz): 434M; Bandwidth (Hz): 1M

UHD: USRP Sink — Device Addr: addr=...168.10.2; Mb0: Subdev Spec: A:0; Samp Rate (Sps): 1M; Ch0: Center Freq (Hz): 434M; Ch0: Gain (dB): 0

Throttle — Sample Rate: 1M

Channel Model — Noise Voltage: 100m; Frequency Offset: 0; Epsilon: 1; Taps: 1+1j; Seed: 0

QT GUI Frequency Sink — Name: ; FFT Size: 1.024k; Center Frequency (Hz): 434M; Bandwidth (Hz): 1M

UHD: USRP Source — Device Addr: addr=...168.10.2; Mb0: Subdev Spec: A:0; Samp Rate (Sps): 1M; Ch0: Center Freq (Hz): 434M; Ch0: Gain (dB): 0

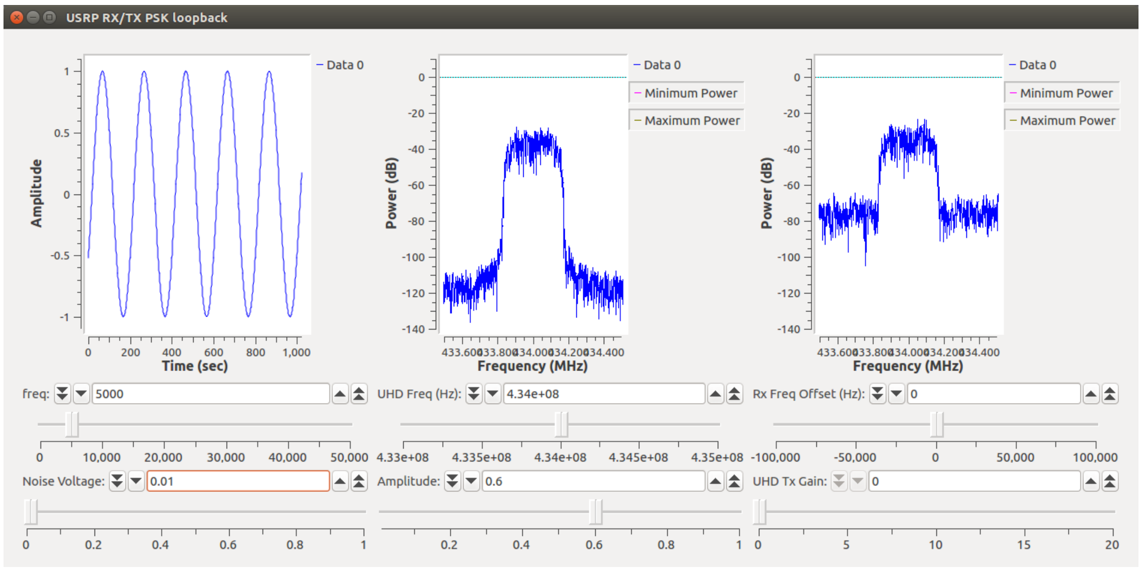
PSK Demod — Number of Constellation Points: 4; Differential Encoding: Yes; Samples/Symbol: 4; Excess BW: 350m; Frequency BW: 62.8m; Timing BW: 62.8m; Phase BW: 62.8m; Gray Code: Yes

Packet Decoder — Access Code: ; Threshold: -1

QT GUI Time Sink — Name: QT GUI Plot; Number of Points: 1.024k; Sample Rate: 1M

Figure 2-7 GNU Radio Flow Graph for the PSK modulation test

In this example we work in the controlled environment previously mentioned, i.e. the *Channel Model* block connects the RX and TX graphs. Alternatively, the graph could be run over a real radio channel using the USRP devices. To achieve this, the *USRP Sink* and *USRP Source* blocks (greyed-out in Figure 2-7) should be enabled, while the *Throttle* and *Channel Model* blocks – disabled. Executing the test in GNU Radio results in an output window presented in Figure 2-8.



USRP RX/TX PSK loopback

freq: 5000

Noise Voltage: 0.01

UHD Freq (Hz): 4.34e+08

Amplitude: 0.6

Rx Freq Offset (Hz): 0

UHD Tx Gain: 0

Figure 2-8 Output of the PSK modulation test

All examples presented so far operated on continuous streams of data samples and used standard GNU Radio blocks only. The next step involves extending the flow graph to provide support for discrete, asynchronous communication in the form of packets. Figure 2-9 presents an example of how packet handling can be achieved using blocks specifically written for the RESCUE project: RESCUE Packet Framer and RESCUE Soft Deframer. The specification of the RESCUE frame format was taken from D3.2. Therefore, these two block constitute the connection to the data link and higher layers, which are being developed in WP3.

Note that on the flow graphs the interfaces operating on packets are grey, whereas interfaces operating on streams are coloured (e.g., the "PDU to Tagged Stream" module in Figure 2-9 converts packets to streams, hence the difference in the colouring of its interfaces).

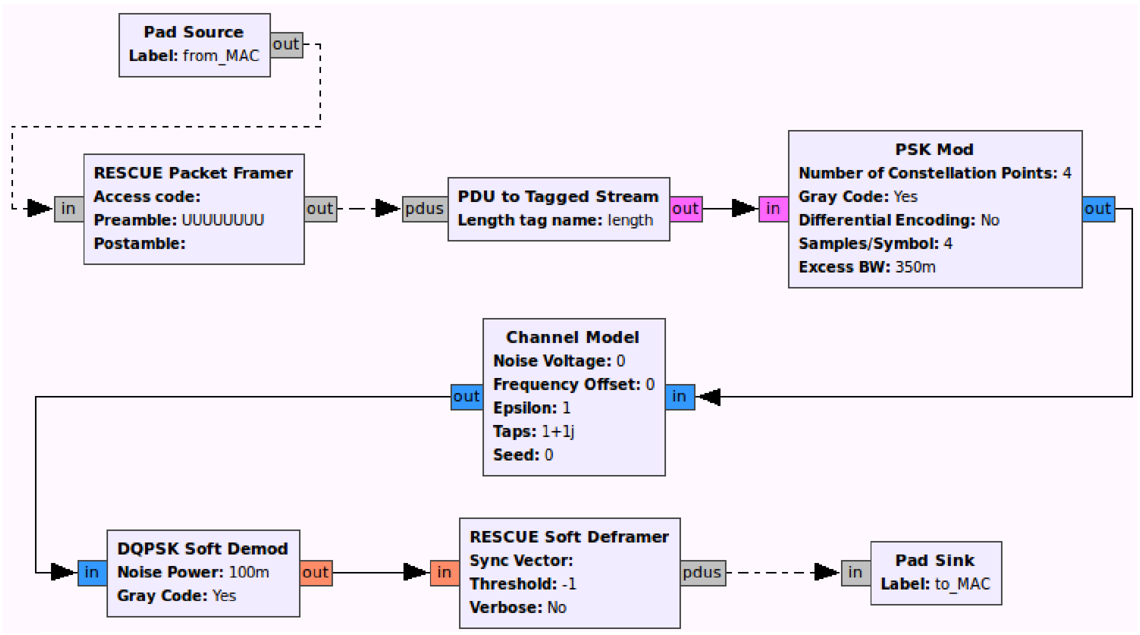


Figure 2-9 Packet-handling using the following blocks: RESCUE Packet Framer, PDU to tagged stream, and RESCUE Soft Deframer.

2.2 RESCUE Source Node

In this section we describe the SDR implementation of a RESCUE source node. Figure 2-10 provides the flow graph of the source node and Table 2-1 provides a description (functionality, input and output) of the flow graph blocks. Note that the blocks that developed within the RESCUE project are denoted as such. Those blocks will be described in detail in the following sections. The remaining blocks are standard GNU Radio blocks.

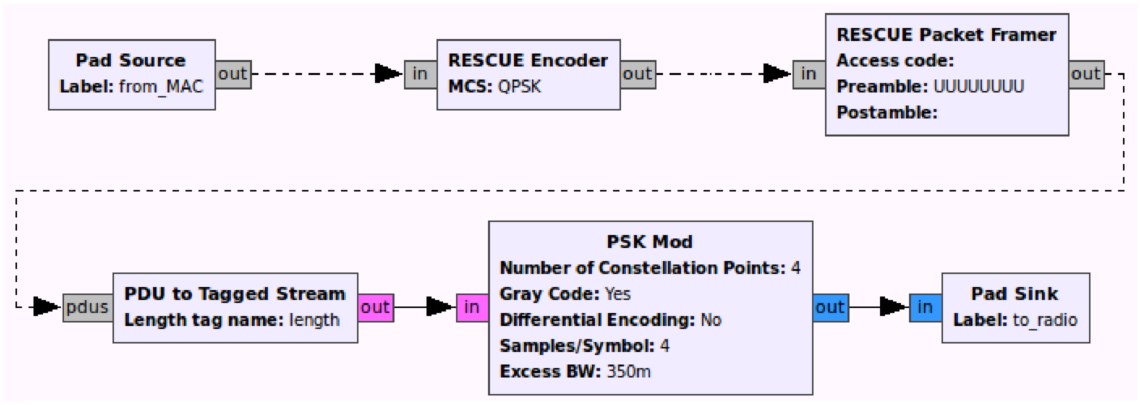


Figure 2-10 RESCUE TX Flow

Table 2-1 RESCUE TX Blocks

Block Name	**Functionality**	**Input**	**Output**
RESCUE Encoder	Encodes the RESCUE PPDU coming from the MAC Layer Uses the specified MCS (Modulation and Coding Scheme)	RESCUE PPDU (PHY Layer Protocol Data Unit)	Encoded RESCUE PPDU
RESCUE Packet Framer	Stamps the incoming message with a preamble, postamble, access code, etc. for synchronization. Parameters: Access code (AKA sync): array of altering 0's and 1's (blank means default)	RESCUE PPDU	RESCUE PPDU with preamble and access code.
PDU To Tagged Stream	Converts received PDUs into a tagged stream of items (standard GNU Radio block)	RESCUE PPDU	Stream of bytes
PSK Mod	Converts the stream of bytes to a radio signal (waveform) with a given number of samples per symbol. (standard GNU Radio block)	Stream of bytes	Radio signal – Complex modulated signal at baseband

The signal flow at the RESCUE source node is described in the following. The RESCUE Encoder block receives PHY layer Protocol Data Units (PPDUs), formatted according to the specification of D3.2, and encodes them. The RESCUE Encoder is a PPDU-oriented block acting as a wrapper for the interleaving and convolutional encoding operation. The block accepts a modulation coding scheme (MCS) parameter set as an input parameter. The header of the PPDU is interleaved according to a fixed pattern and the payload is interleaved using a random pattern. The interleave operation uses the `trellis::interleaver` class provided by GNU Radio. After interleaving, the frame is encoded according to the given MCS using the `encode_single()` method developed in WP2. From the output of the RESCUE Encoder the PPDU is passed to the RESCUE Packet Framer, which extends the PPDU with a preamble, postamble, and the access code used to facilitate the symbol level synchronization. The PPDU is then converted to a tagged stream of bytes using a standard GNU Radio PDU To Tagged Stream block. Finally, the byte stream is converted to baseband QPSK modulated samples using the PSK Mod block (part of the GNU Radio standard blocks).

2.2.1 Source and Channel Coding

In the implementation of the RESCUE Encoder, each node performs a simple concatenated convolutional encoding which is illustrated in Figure 2-11 and described in Table 2-2. A binary source sequence $\mathbf{u}$ is first interleaved by a random interleaver Π_1 whose role is to enable the iterative decoding process to utilize the correlation knowledge, since decoders exchange information via the interleaver/deinterleaver based on the turbo principle. As described in Section 2.3.6, the decoded soft information of $\mathbf{u}$ is exchanged among the decoders and hence the interleaver Π_1 is needed. The interleaved sequence is then encoded by a convolutional encoder C resulting in a coded sequence. It is further interleaved by another interleaver Π_2, the length of which depends on the coding rate of the encoder C. After that, the interleaved version of the coded sequence is doped-and-accumulated by a so-called accumulator (ACC) with a doping ratio P_d. Finally, the modulated symbol sequence generated by the modulator for the doped-accumulated binary sequence is transmitted to the destination node over an additive white Gaussian noise (AWGN) channel. In the implementation, we consider three kinds of modulation schemes, which are binary phase-shift keying (BPSK), quadrature phase-shift keying (QPSK) and 16-quadrature amplitude modulation (16QAM).

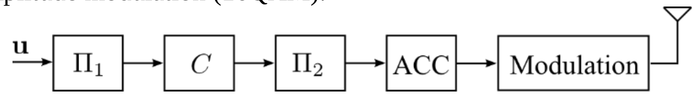


Figure 2-11 Structure of the RESCUE Encoder

For simplicity, we assume that the channels from each node to the destination node are orthogonal to each other. Therefore, the received signal $\mathbf{y}_k$ at the destination from the k-th node is simply expressed as:

$$\mathbf{y}_k = \mathbf{s}_k + \mathbf{w}_k, \tag{2.1}$$

where $\boldsymbol{w}_k$ is the complex white Gaussian noise sequence with the variance per dimension σ^2 and $\mathbf{s}_k$ is the modulated symbol sequence.

The motivation of using this code structure is twofold: 1) the implementation of the convolutional code is simple; and 2) combing ACC and modulation provides an additional degree of freedom in finding good coding parameters to improve the system performance.

Table 2-2 Implementation of the RESCUE Encoder

Item	Function	Input	Output	Implementation
Random interleavers Π_1, Π_2	`interleave()`	1) Bit sequence (integer vector), 2) permutation order (integer vector)	Interleaved bit sequence (integer vector)	Use `gen_intrlv_random()` function, which is implemented by `rand()` and `quick_sort()` functions, to generate permutation order, and then function `interleave()` simply change the order of input sequence.
Encoder C	`nrc_encode()`	1) generator polynomial (integer array), 2) data sequence (integer vector), 3) encoded sequence (integer vector), 4) coding rate, e.g., ½, input 2 (integer), 5) memory size (integer)	void	Function `oct2dec()` convert the generator polynomial in octal form, e.g., [7, 5] into that in decimal form; and then `nrc_encode()` function encodes the input sequence using bitwise operator.
ACC	`ACC()`	1) data sequence (integer vector), 2) doping ratio, (integer)	doped-and-accumulated data sequence (integer vector)	Use modulo-2 (% 2) operation to generate the encoded sequence, and then replace the input bit by encoded bit according the doping ratio.
BPSK Modulation	---	----	----	Simply convert 1 to 1 as well as 0 to -1, and store the results into a float vector.
QPSK, 16QAM Modulation	`mapper_QPSK()`, `mapper_16QAM()`	1) bit sequence (integer vector), 2) real part and 3) imagination part of modulated signal (float vector), 4) mapping table (integer vector)	void	Convert two bits (QPSK) or four bits (16QAM) into a decimal number, and then associate the corresponding soft symbol based on the mapping table and the decimal number.

Note. The libraries using in the C++ implementation are basic, including `<vector>`, `<time.h>`, `<stdlib.h>` and `<cmath>`.

2.2.2 Modulation and Coding Schemes

For the modulation scheme, BPSK is insufficient for supporting flexible coding rates. For example, when the channel condition is sufficiently good, high rate communication can be guaranteed. In addition, with the purpose of making fair comparison with other techniques, higher order modulation, e.g., QPSK, 16QAM, is also considered in the implementation of the RESCUE Encoder. As stated above, by combining the ACC encoder and modulation scheme, we can further search for good coding parameters including the doping ratio of ACC and mapping rule of the modulation to improve the system performance.

The extrinsic information transfer (EXIT) chart analysis is conducted for obtaining good coding parameters in the implementation. However, it should be emphasized that such a method cannot guarantee the optimality of the obtained parameters. Also, this method assumes that the demapper/demodulator has feedback information from the decoder of the outer code (convolutional code), i.e., the demapper/demodulator is performed iteratively. Based on the EXIT chart analysis, which are shown in Figure 2-12 EXIT chart of QPSK and Figure 2-13 EXIT chart of 16QAM, we obtained a series of parameters used in the coding scheme. The parameters are summarized in Table 2-3.

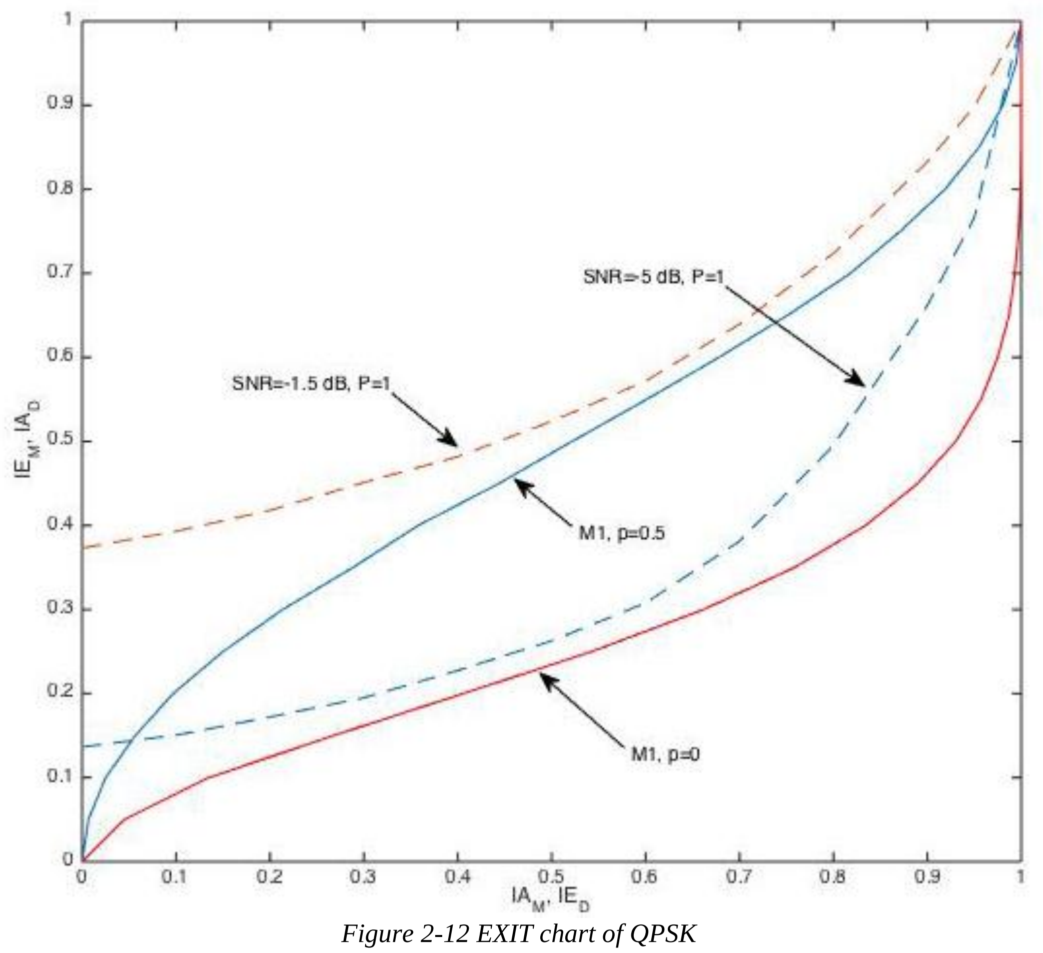


Figure 2-12 EXIT chart of QPSK

Table 2-3 Modulation and coding scheme parameters

Modulation	Doping ratio	Mapping rule
QPSK	8	natural mapping
16QAM	1	Modified set partitioning (MSP) mapping

In the conventional communication systems, Gray mapping is commonly used since it minimized the bit error rate when the feedback is not available at the demapper. However, when the feedback, which is usually from the decoder, is available, Gray mapping is not the optimal choice. In this case, natural mapping/MSP mapping is used to improve the BER performance.

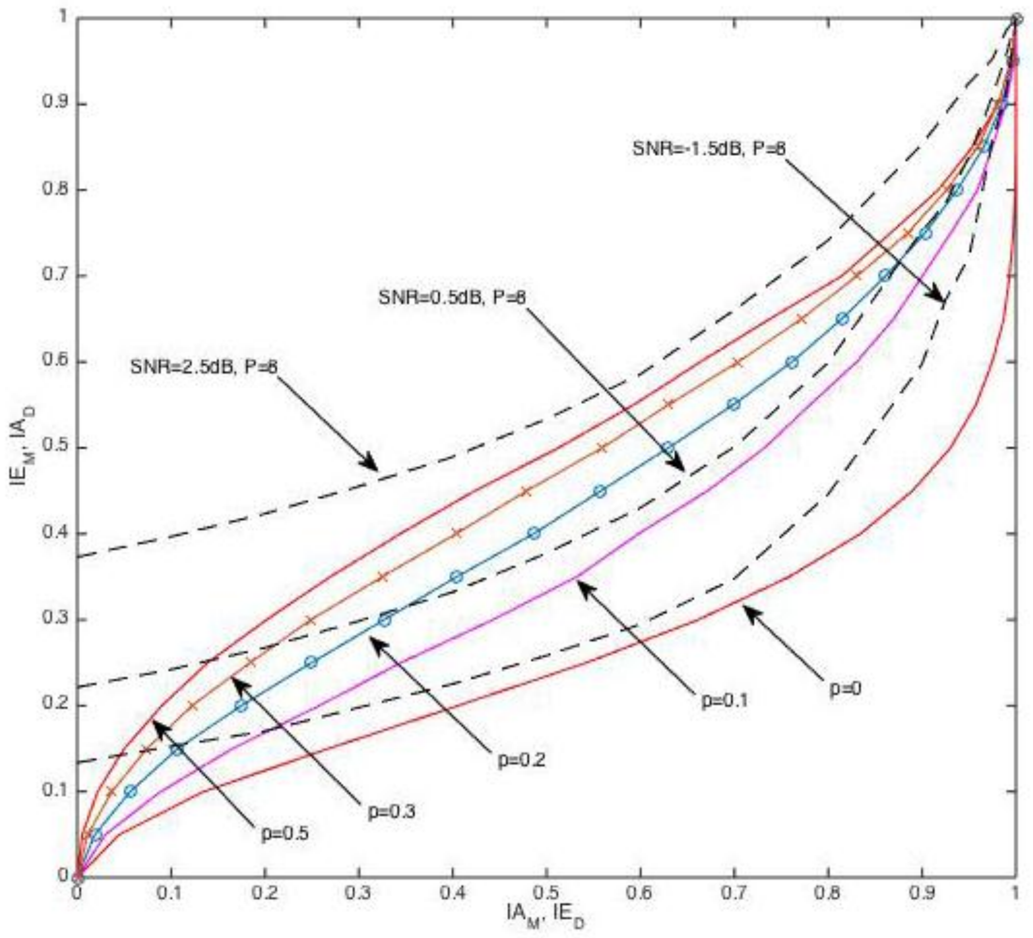


Figure 2-13 EXIT chart of 16QAM

Three Modulation and Coding Schemes were implemented in RESCUE software:

- QPSK Payload
- 16QAM Payload
- PHY Header - same for QPSK and 16QAM

The parameter sets of each scheme are summarized in Table 2-4.

Table 2-4 parameters of the Modulation and Coding Scheme implemented in RESCUE

	PHY Header	**QPSK Payload**	**16QAM Payload**
Overall Rate	1/4	1/2	3/4
Inner Code	**ACC**	**doped ACC with puncturing**	**doped ACC with puncturing**
Doping	N/A	8	1
Puncture Pattern	N/A	[0 1]	[0 1]
Inner Code Rate (after puncturing)	1/2	1/1	1/1
Memory	1	1	1
Outer Code	**CC**	**CC**	**CC with puncturing**
Polynomial Description	{3,2}	{3,2}	{7,5}
Puncture Pattern	N/A	N/A	[1 0 1] for even positions [1 1 0] for non-even
Outer Code Rate (after puncturing)	1/2	1/2	3/4
Memory	1	1	2
Constellations	**4 or 16**	**4**	**16**

Mapping Rule	Gray Code	Gray Code	Gray Code

2.2.3 RESCUE Packet Framer

The RESCUE Packet Framer is responsible for providing physical header protection while supporting various payload lengths. Even though joint decoding allows reliable communication in a low SNR regime, the PHY header is unique for each copy of the frame and, therefore, cannot be jointly decoded. Furthermore, the header needs to be decoded without errors since it contains critical information which is needed to decode the payload. Consequently, the header should be protected with a stronger code than the payload.

The choice of the code rate for the header depends on how many copies will be jointly decoded. To enable the gains of RESCUE, we should be able to decode the header at a lower SNR level error-free in comparison to the payload. On the other hand, it is not desirable to decrease the code rate too much since it decreases the overall bit rate and increases decoding complexity. In Figure 2-14, an example for depicting the FER of PHY header and joint decoded payload in AWGN channel is shown. The code rates for the header and payload are ¼ and ½, respectively. It can be seen that in this scenario, we can utilize the gain from two copies of the payload. The third copy is irrelevant because the SNR regime where it would be beneficial is too low in order to receive the header correctly.

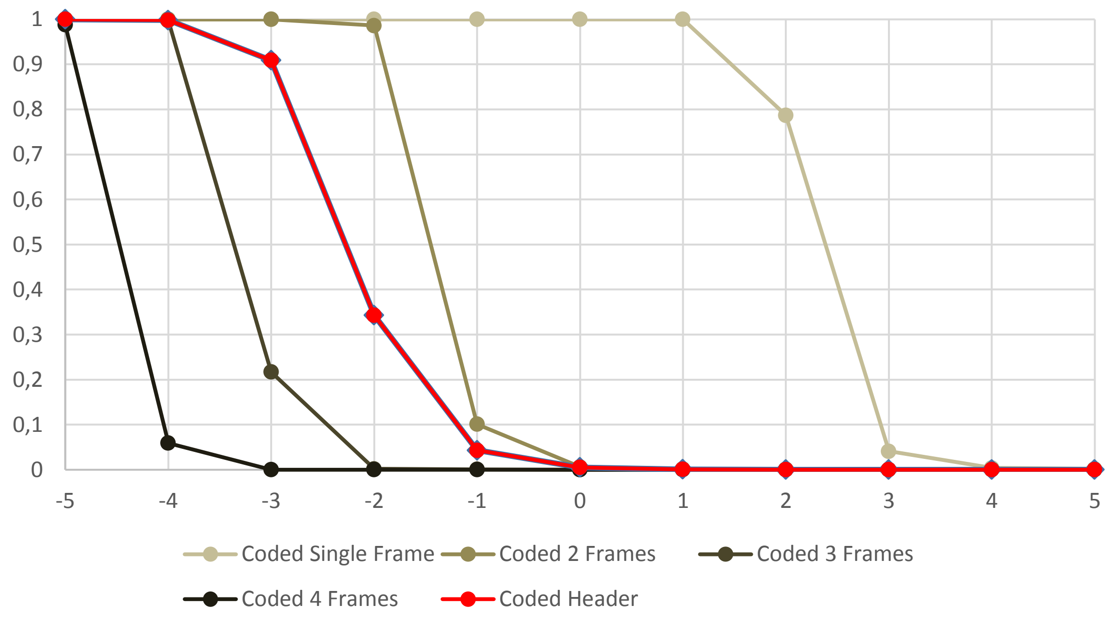


Figure 2-14 Frame Error Rate vs SNR (dB) in QPSK

Another practical parameter to be considered is the length of the payload. For example, in low power machine-to-machine (M2M) communications, the length of the payload can be less than 127 bytes [anton2014machine]. On the other hand, even though small frame sizes are also favoured in vehicle-to-vehicle (V2V) communications, the maximum transmit unit is usually set to 1500 bytes. The maximum size of the RESCUE DATA frame payload is set to 2328 bytes in [RESCUE D3.2]. Therefore, we will simulate RESCUE with various payload sizes.

It is intuitive that increasing the length of the payload increases FER. However, decreasing the length of the payload increases the power consumption per information bit. The energy per payload information bit can be calculated as:

$$\left(\frac{E_b}{N_0}\right)^{payload} = \frac{\frac{1}{M}\left(\frac{L^{header}}{R_c^H} + \frac{L^{payload}}{R_c}\right) \times SNR}{L^{payload}} \tag{2.2}$$

where M denotes the number of bits per modulation symbol, L^{header} and $L^{payload}$ are the lengths of the uncoded header and payload, respectively, and R_c^H and R_c are the code rates for the header and payload, respectively. FER versus $(E_b/N_0)^{payload}$ for two coded frames and for various coded payload lengths is plotted in Figure 2-15. Other parameter values are: M=2 (QPSK), $L^{header} = 31$ as defined in [RESCUE D3.2] and the code rates for the header and the payload are ¼ and ½, respectively, as in Figure 2-14. It can be seen that when increasing the payload size larger than 1500 bytes, the gain is almost negligible. However, decreasing the payload size from 1000 to 100 bytes the loss is roughly 3 dB.

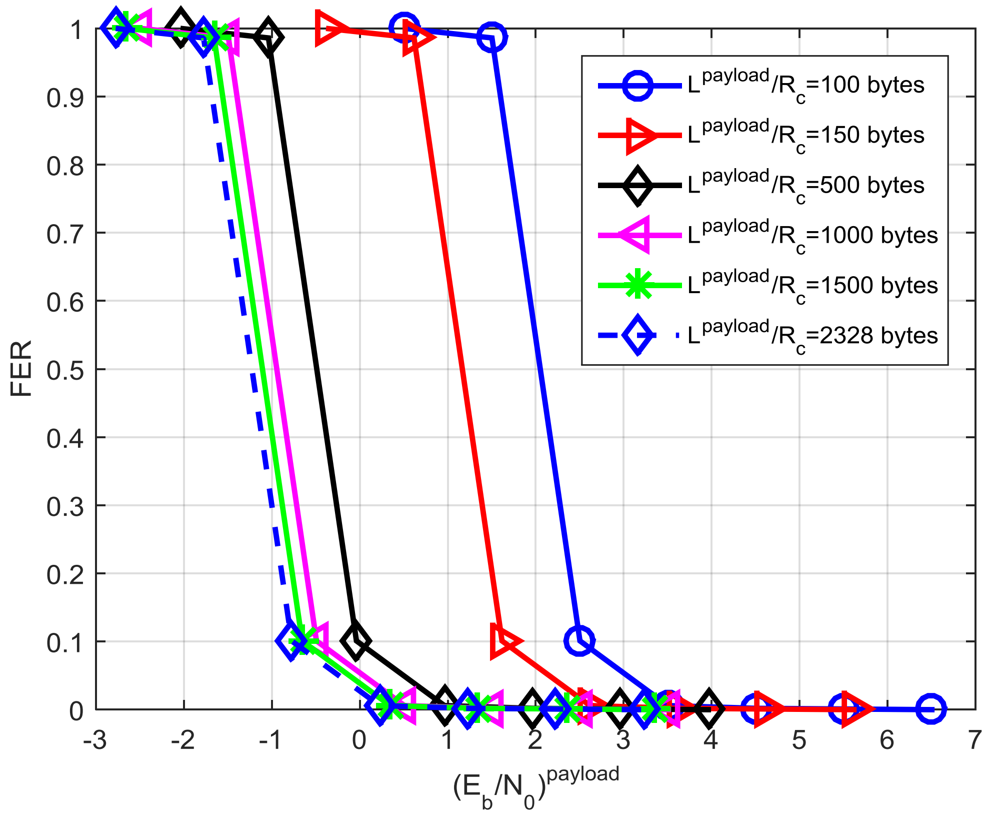


Figure 2-15 Frame Error Rate vs $(E_b/N_0)^{payload}$ (dB) in QPSK for two coded frames

2.3 RESCUE Destination Node

2.3.1 Radio Signal Reception

In this section we explain what necessary actions must be taken in order to enable radio reception (RX) with USRP and GNU Radio. Figure 2-16 presents the flow graph of the RX flow in a RESCUE node and Table 2-5 provides RX block descriptions. In the table, italics denotes blocks that are developed within the RESCUE project. These blocks will be described in detail below. Evidently on the receiver side all block are developed within the RESCUE project.

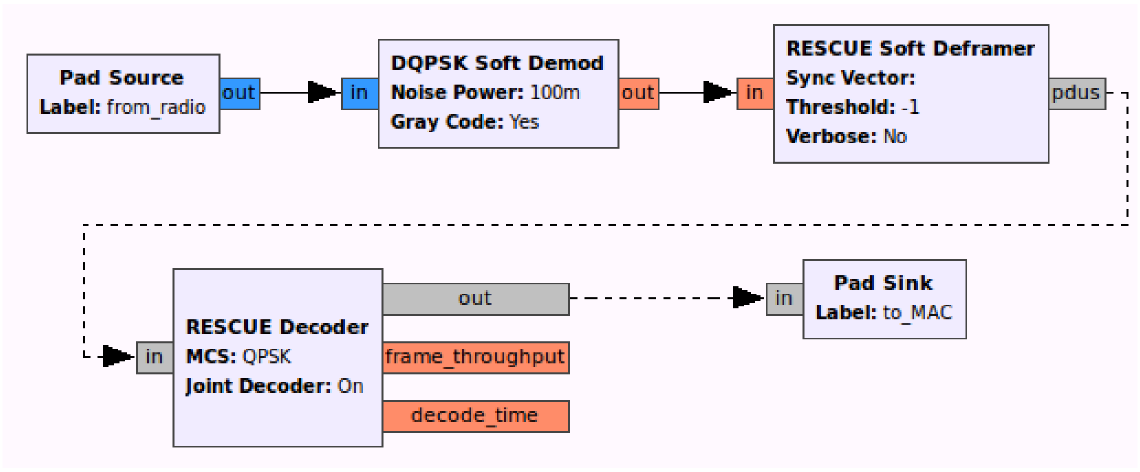


Figure 2-16 RESCUE RX Flow

Table 2-5 RESCUE RX blocks

Block Name	**Responsibility**	**Input**	**Output**
DQPSK Soft Demod	Converts the radio signal (e,g., coming from USRP device) into soft bits (LLR stream). Parameters: `noise_power` - specifies the noise variance used in LLR calculation	Complex modulated signal at baseband.	"Soft-bits". LLR stream, one LLR per bit.
RESCUE Soft Deframer	Converts "continuous" LLR stream to messages by matching (correlating) the access code and sends them to the output port. Parameters: `Access code` (AKA `sync`): array of altering 0's and 1's (blank means default array – 64 bits long) `Threshold`: Number of bits that can be wrong (-1 means default). Default is 12.	LLR stream	RESCUE PPDU
RESCUE Decoder	Decodes the RESCUE PPDU. Supports joint decoding: Frames (PPDUs) with failed PHY header check are dropped. All other frames are forwarded, regardless if the payload CRC check is successful or not. Frames with failed CRC check are stored for later joint-decoding. Uses the specified MCS.	Encoded RESCUE PPDU	Uncoded RESCUE PPDU

The blocks *DQPSK Soft Demod* and *RESCUE Soft Deframer* work on soft-bits (LLRs) and are counterparts of the standard GNU Radio blocks that work on (hard) bits. These blocks are universal, reusable GNU Radio blocks, so they could be used also in other projects.

DQPSK Soft Demod is an example. Additionally, three other soft-demodulators were developed for RESCUE: QPSK and 16QAM/D16QAM. Moreover OFDM and GMSK demodulators are currently under development.

The decision-directed carrier phase recovery method can cause the phase-ambiguity problem, which is a consequence of the rotational symmetry of the constellation. As a result, a carrier tracking PLL locks to a carrier with a phase offset. In order to counteract this phase-ambiguity phenomenon either a differential encoding is used or the received signal is de-rotated with the aid of the unique symbol sequence passed to the data stream. Since the differential encoding penalizes the noise resilience of the demodulation scheme the unique word method is used in RESCUE.

The solution for the phase-ambiguity problem applied in RESCUE is based on the idea of using the `access_code` (AKA as `sync`) as a training sequence. The phase is set correctly at the frame start. This solution works only if the PLL does not get out of order during the frame transmission. It works only for high SNR values. In case of low SNR values, which is actually the RESCUE case, the PLL gets out of order very often, even during the frame transmission. This solution can be represented by the flow graphs in Figure 2-17.

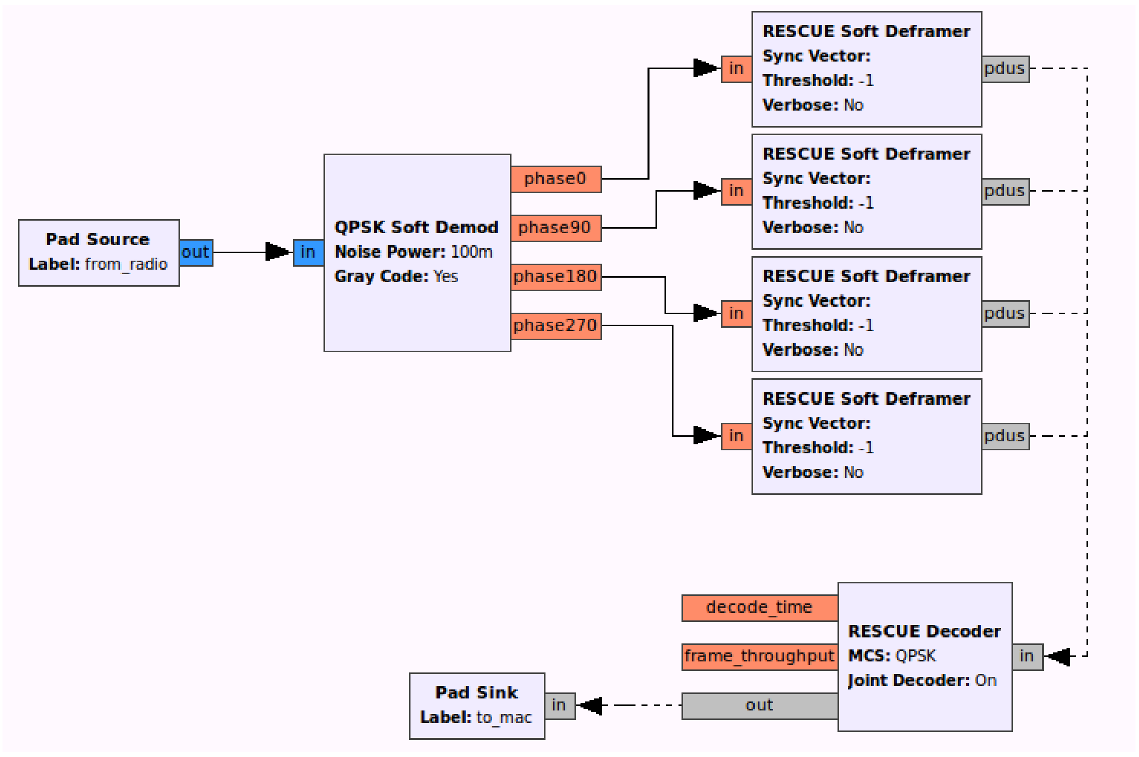


Figure 2-17 RESCUE RX Flow for non-differential modulations

In conclusion, this solution will work for high SNR values only. Therefore, it cannot be recommended for RESCUE. To enable QPSK for lower SNR values, an additional “training sequence” should be sent inside the frame. However, this requires more resources. It is planned to introduce a training sequence to the GNU Radio RESCUE flow graph in the integration phase of the RESCUE project in WP4.

2.3.2 Demodulator

DQPSK Soft Demod is a hierarchical block that demodulates the signal using the flow graph shown in Figure 2-18. The received signal is passed to the input of the AGC2 block which performs the automatic gain control loop with an adjustable attack and decay rate. The input signal is scaled to the unity power. The AGC2 block is a part of the standard GNU Radio distribution. The scaled version of the received signal is passed to the input of the FLL band edge block which performs coarse carrier synchronization. The block is a second order frequency locked loop with an error signal resulting from filtering the upper and lower band edges. After coarse frequency synchronization, the received signal is resampled using a Polyphase Clock Sync block which is an implementation of a filter bank-based timing synchronizer implemented according to the algorithm described in [harris_01]. The resampled signal is passed to the input of the constellation receiver block which is an implementation of the Costas loop-based decision directed receiver. The stream of received symbol estimates generated by the constellation receiver is passed to the Constellation Soft Decoder aimed to convert them to the stream of bit LLR values based on the look-up-table (LUT) which depends on the noise-power. For differential modulations, the diff-decoder function, e.g. (s1-s0) modulo 4, has to be done on bit probabilities (LLRs). This is done inside the soft differential decoder block using fuzzy logic.

Note, that the LLR stream is passed from the soft demodulator to the decoder through the RESCUE Soft Deframer GNU Radio block, which is described in detail in Section 2.3.7. The RESCUE Soft Deframer (RSD) operates on soft bits (LLRs) instead of “hard” bits. Hard bits appear only at the output of the decoder, never before. First, the RSD search for the Frame Access Code. If the code is found with a BER lower than 0.1875 than the RSD starts to decode the Header, and verify its checksum. If the header checksum is correct then the payload is transferred to the RESCUE decoder. After decoding, the whole decoded frame is transferred to higher layers.

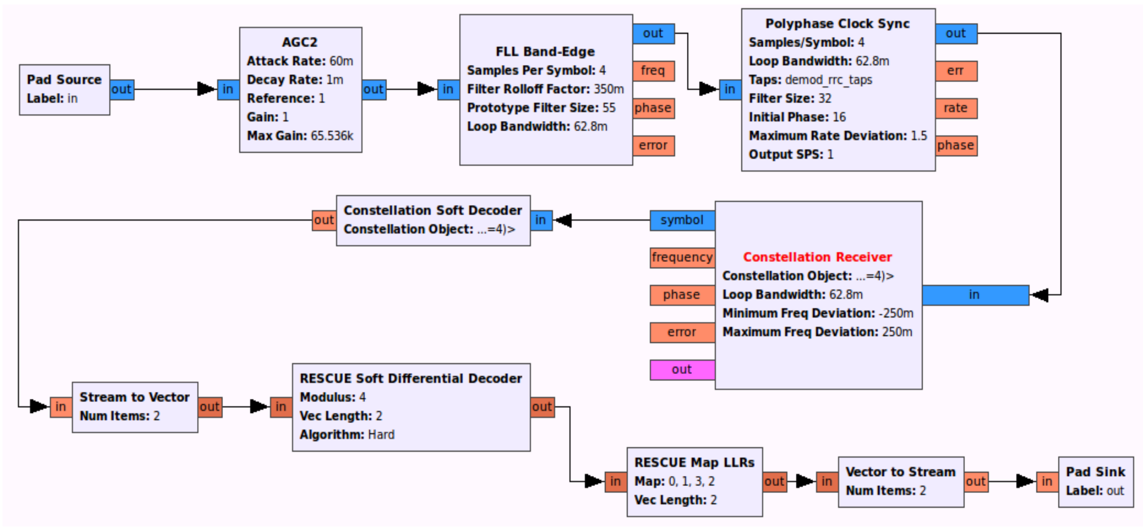


Figure 2-18 DQPSK Soft Demodulation flow

2.3.3 Demapping Process

In this section we describe methods for performing the demapping process. The methods described in Sections 2.3.3.1-0 were implemented and validated but not integrated in the SDR platform. The main obstacle was the lack of a feedback loop implementation in GNU Radio and the training sequence. This issues have not yet been resolved. The integration of general LLR calculation and SNR estimation methods with GNU Radio has been moved to Task 4.3 of WP4.

2.3.3.1 General LLR Calculation

Consider M-ary modulation with constellation $[s_1, s_2, \dots, s_M]$. Let the binary code word associated with the symbol s_i be denoted as $b^i = \left[b_1^i, b_2^i, \dots, b_{log_2 M}^i\right]$. Let the received noisy symbol be denoted as r. We want to calculate the LLRs for the unknown code word $\left[x_1, x_2, \dots, x_{log_2 M}\right]$. Assuming that the noise follows Gaussian distribution with variance σ^2, the extrinsic LLR of bit$x_j$from the soft demapper is given by:

$$L^E(x_j) = ln \frac{\sum_{\substack{i=1 \\ b_j^i=1}}^{M} e^{\frac{-|r-s_i|^2}{2\sigma^2} + \sum_{k\neq j}^{log_2 M} lnP^A\left(x_k=b_k^i\right)}}{\sum_{\substack{i=1 \\ b_j^i=0}}^{M} e^{\frac{-|r-s_i|^2}{2\sigma^2} + \sum_{k\neq j}^{log_2 M} lnP^A\left(x_k=b_k^i\right)}} \tag{2.3}$$

with the *a priori* feedback:

$$\begin{gathered} \mathrm{lnP^A(x_k = 1) = L^A(x_k) - ln}\left(1 + \mathrm{e}^{\mathrm{L^A(x_k)}}\right) \\ \mathrm{lnP^A(x_k = 0) = -ln}\left(1 + \mathrm{e}^{\mathrm{L^A(x_k)}}\right) lnP^A(x_k = 1) = L^A(x_k) - ln\left(1 + e^{L^A(x_k)}\right) \\ lnP^A(x_k = 0) = -ln\left(1 + e^{L^A(x_k)}\right) \end{gathered} \tag{2.4}$$

where $L^A(x_k)$ is the *a priori* LLR given by the decoder. After combining (2.3) and (2.4) and simple mathematical transformations the equation (2.3) can be rewritten as

$$L^E(x_j) = ln \frac{\sum_{\substack{i=1 \\ b_j^i=1}}^{M} e^{\frac{-|r-s_i|^2}{2\sigma^2} + \sum_{k\neq j}^{log_2 M} b_k^i L^A(x_k) - \sum_{k\neq j}^{log_2 M} ln\left(1+e^{L^A(x_k)}\right)}}{\sum_{\substack{i=1 \\ b_j^i=0}}^{M} e^{\frac{-|r-s_i|^2}{2\sigma^2} + \sum_{k\neq j}^{log_2 M} b_k^i L^A(x_k) - \sum_{k\neq j}^{log_2 M} ln\left(1+e^{L^A(x_k)}\right)}} \tag{2.5}$$

Since the term $\sum_{k\neq j}^{log_2 M} ln\left(1 + e^{L^A(x_k)}\right)$ appears in every addend in both numerator and denominator it can easily be eliminated and the equation (2.5) reduces to:

$$LLR(x_j) = ln \frac{\sum_{\substack{i=1 \\ b_j^i=1}}^{M} e^{\frac{-|r-s_i|^2}{2\sigma^2} + \sum_{k\neq j}^{log_2 M} b_i^k L^A(x_k)}}{\sum_{\substack{i=1 \\ b_j^i=0}}^{M} e^{\frac{-|r-s_i|^2}{2\sigma^2} + \sum_{k\neq j}^{log_2 M} b_i^k L^A(x_k)}} \tag{2.6}$$

The Jacobian logarithm can be used to simplify the implementation. The Jacobian logarithm can be defined with the following recursive rule:

$$\Upsilon(L_1, L_2, \dots, L_N) = \Upsilon\left(L_1, \Upsilon(L_2, \dots, L_N)\right) \tag{2.7}$$

where:

$$\begin{gathered} \Upsilon(L_1, L_2) = max(L_1, L_2) + ln\left(1 + e^{-|L_1 - L_2|}\right) \\ \Upsilon(L_1) = L_1 \end{gathered} \tag{2.8}$$

which can be further approximated as:

$$\Upsilon\left(L_1, L_{2,} \dots, L_N\right) = max\left(L_1, L_{2,} \dots, L_N\right). \tag{2.9}$$

Since the Jacobian logarithm possess the important property

$$\Upsilon(L_1, L_2, \dots, L_N) = ln\left(\sum_{n=1}^{N} e^{L_n}\right) \tag{2.10}$$

equation (2.6) can be written as:

$$\begin{aligned} LLR(x_j) = {} & max_{\substack{i=1 \\ b_j^i=1}}^{M} \left(\frac{-|r-s_i|^2}{2\sigma^2} + \sum_{k\neq j}^{log_2 M} b_i^k L^A(x_k)\right) \\ & - max_{\substack{i=1 \\ b_j^i=0}}^{M} \left(\frac{-|r-s_i|^2}{2\sigma^2} + \sum_{k\neq j^{log_2 M}} b_i^k L^A(x_k)\right) \end{aligned} \tag{2.11}$$

The calculation of the LLR value can be further simplified using some properties of the $\frac{-|r-s_i|^2}{2\sigma^2}$ expression. If we denote $\Re(\mathrm{r})$ as r_x and $\Im(\mathrm{r})$ as r_y we can write:

$$\frac{-|r-s_i|^2}{2\sigma^2} = \frac{-(r_x - x_i)^2 - \left(r_y - y_i\right)^2}{2\sigma^2} \tag{2.12}$$

After refactoring the equation (10) we may rewrite it as:

$$\frac{-|r-s_i|^2}{2\sigma^2}=\frac{-\left(r_x+r_y\right)^2-x_{\mathrm{i}}^2-y_i^2+2x_ir_x+2y_ir_y}{2\sigma^2} \tag{2.13}$$

One can easily observe that the term $\frac{\left(r_x+r_y\right)^2}{2\sigma^2}$ appears in every opperand of the $max(\quad)$ operators. Since the resulting LLR value is a difference of two $max(\quad)$ operations it has no influence of the resulting LLR and thus can be removed. Additionally the term $\frac{(x_i^2+y_{\mathrm{i}}^2)}{2\sigma^2}$ can be precalculated at the stage of construction of the demapper object. Denoting $\frac{(x_i^2+y_i^2)}{2\sigma^2}$ as A_{i} the whole calculation can be written as:

$$\begin{aligned} LLR(x_j)=max_{\substack{i=1\\b_j^i=1}}^{M}\left(-A_{\mathrm{i}}+\frac{s_{xi}r_x+s_{yi}r_y}{\sigma^2}+\sum_{k\neq j}^{log_2M}b_i^kL^A(x_k)\right)\\ -max_{\substack{i=1\\b_j^i=0}}^{M}\left(\frac{-|r-s_i|^2}{2\sigma^2}+\sum_{k\neq j^{log_2M}}b_i^kL^A(x_k)\right)\end{aligned} \tag{2.14}$$

2.3.3.2 Example LLR Calculation for QPSK

Assume a QPSK with constellation $[s_1,s_2,s_3,s_4]=[00,01,10,11]$. According to the definition (2.3), the extrinsic LLR of the first bit x_1 can be calculated as:

$$LLR(x_1)=ln\frac{e^{\frac{-|r-s_3|^2}{2\sigma^2}+L^A(x_2=0)}+e^{\frac{-|r-s_4|^2}{2\sigma^2}+L^A(x_2=1)}}{e^{\frac{-|r-s_1|^2}{2\sigma^2}+L^A(x_2=0)}+e^{\frac{-|r-s_2|^2}{2\sigma^2}+L^A(x_2=1)}} \tag{2.15}$$

which can be further evaluated to

$$LLR(x_1)=ln\frac{e^{\frac{-|r-s_3|^2}{2\sigma^2}-ln\left(1+e^{L^A(x_2)}\right)}+e^{\frac{-|r-s_4|^2}{2\sigma^2}+L^A(x_2)-ln\left(1+e^{L^A(x_2)}\right)}}{e^{\frac{-|r-s_1|^2}{2\sigma^2}-ln\left(1+e^{L^A(x_2)}\right)}+e^{\frac{-|r-s_2|^2}{2\sigma^2}+L^A(x_2)-ln\left(1+e^{L^A(x_2)}\right)}} \tag{2.16}$$

After application of simple math the equation can be rewritten as

$$LLR(x_1)=ln\frac{e^{\frac{-|r-s_3|^2}{2\sigma^2}}+e^{\frac{-|r-s_4|^2}{2\sigma^2}+L^A(x_2)}}{e^{\frac{-|r-s_1|^2}{2\sigma^2}}+e^{\frac{-|r-s_2|^2}{2\sigma^2}+L^A(x_2)}} \tag{2.17}$$

Using the Jacobian logarithm the equation can be transformed to:

$$\begin{aligned} LLR(x_1)=max\left(-A_3+\frac{s_{x3}r_x+s_{y3}r_y}{\sigma^2},-A_4+\frac{s_{x4}r_x+s_{y4}r_y}{\sigma^2}+L^A(x_2)\right)\\ -max\left(-A_1+\frac{s_{x1}r_x+s_{y1}r_y}{\sigma^2},-A_2+\frac{s_{x2}r_x+s_{y2}r_y}{\sigma^2}+L^A(x_2)\right)\end{aligned} \tag{2.18}$$

2.3.3.3 Performance Evaluation of the Approximated LLR Calculation Method

The approximate LLR calculation was tested on the 16-QAM modulation with MSP mapping. The constellation diagram of the tested modulation scheme is presented in Figure 2-19.

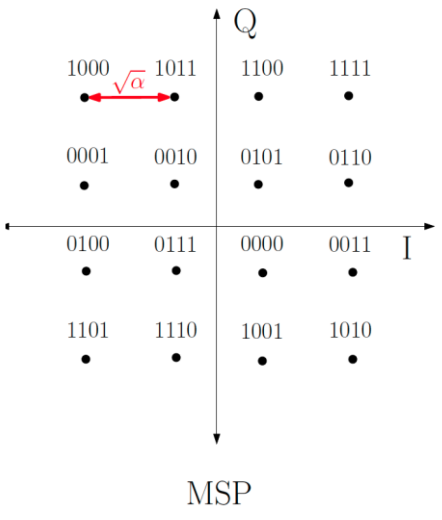


Figure 2-19 Constellation diagram of the reference modulation

The tests were conducted in two stages. In the first stage, an LLR calculation without an *a priori* feedback was examined. During the test a series of random 0-1 bits were generated. Each generated bit was corrupted by white noise with a variance of E_B/N_0. The vector of the *a priori* LLRs was filled with zeros. For each of the E_B/N_0 values the simulations were performed until the reception of 100 erroneous bits. The results of the simulations are presented in Figure 2-20 and Figure 2-21.

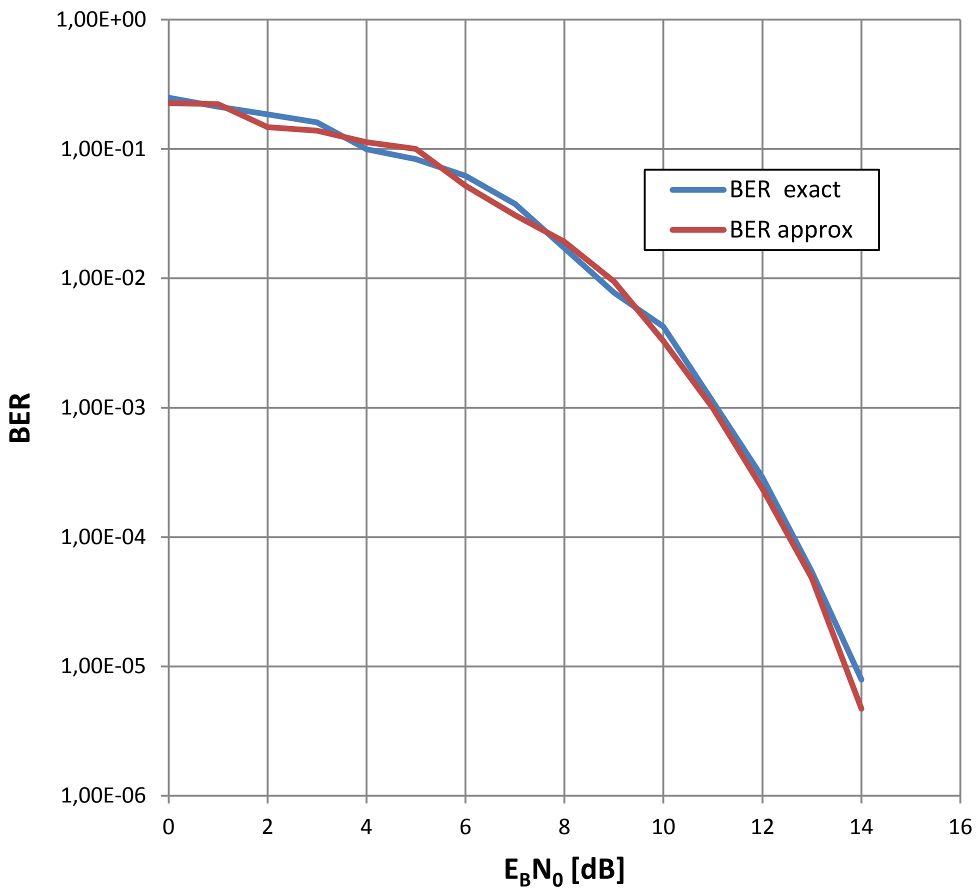


Figure 2-20 BER curve for 16 QAM with MSP mapping

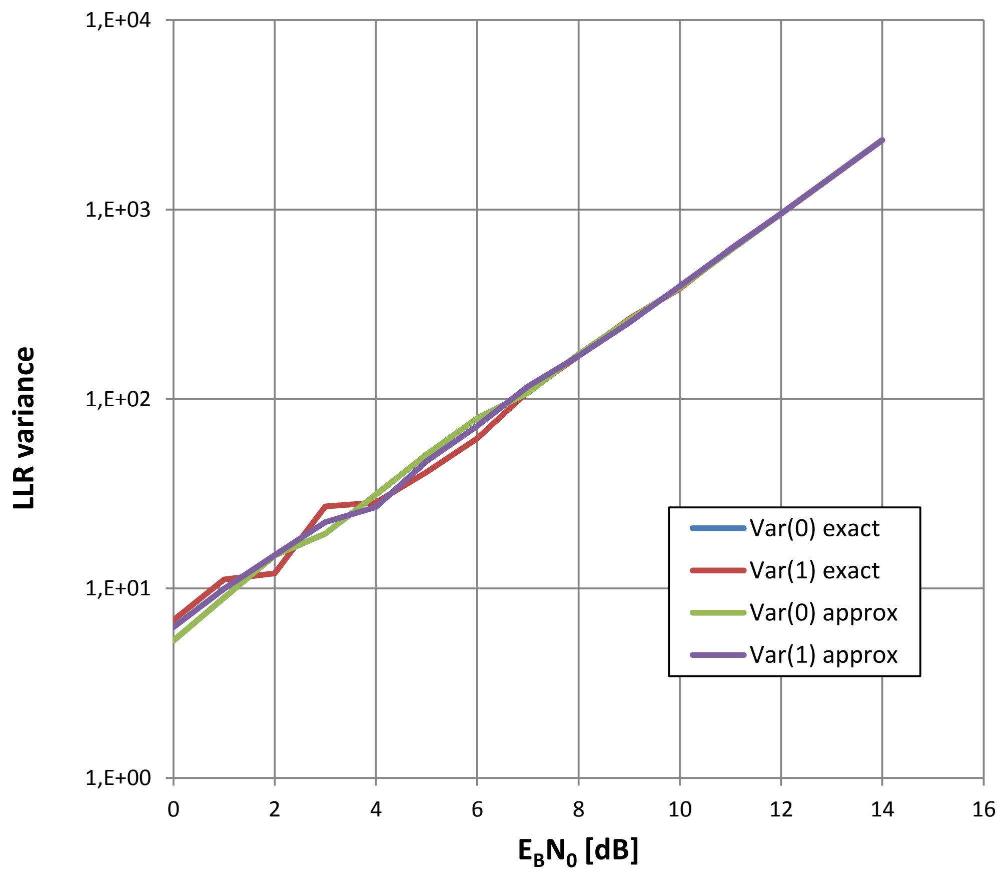


Figure 2-21 LLR variance for 16 QAM with MSP mapping

2.3.3.4 Lookup Table Implementation

Direct symbol remapping using the above described method is computationally expensive. This resulted in slowing the simulation and is difficult to perform in a real-time scenario. GNU Radio allows to speed up decoding symbols by means of the Look Up Table (LUT) technique. The implementation uses the `soft_dec_table()` function of GNU Radio.

The implementation of LUT is presented in Figure 2-22. The constellation decoder implemented by Constellations Soft Decoder takes as a parameter `Constellation Object` which sets up the constellation type. In the Constellation Object (ID: `qpsk2` in Figure 2-22) the `Soft Decision LUT` parameter indicates which type of decoding will be applied (,"None"' means the direct method).

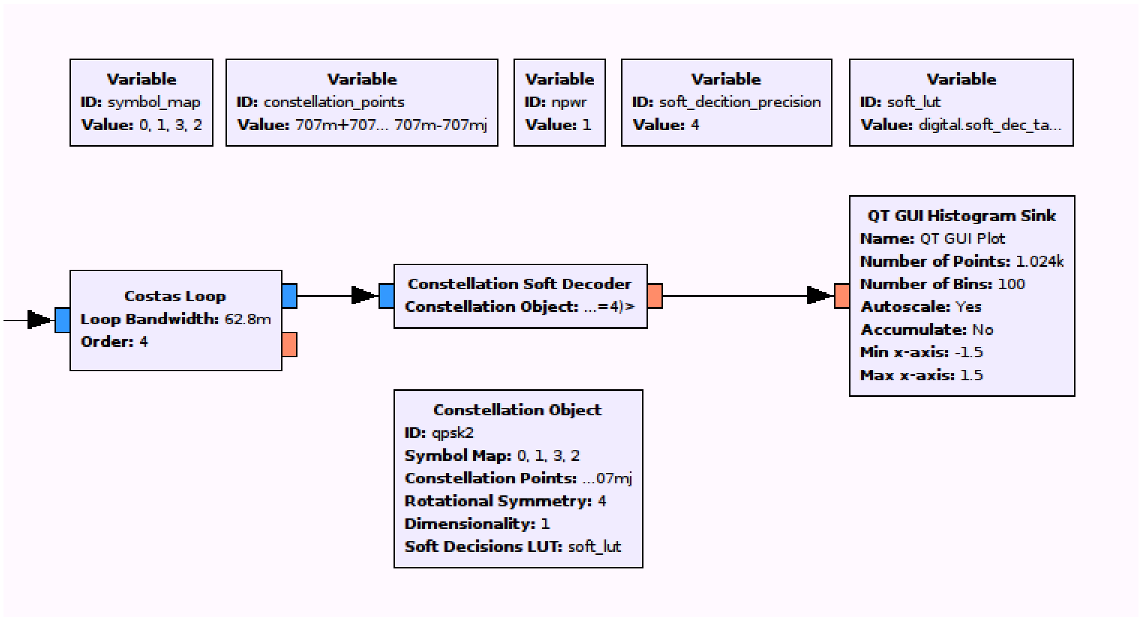
In this implementation, the `soft_dec_table()` function was applied (the `soft_lut` variable in parameter) in the following way:

```
digital.soft_dec_table(constellation_points, symbol_map,
soft_decition_precision, npwr),
```

with the following parameters:

- `constellation_points:` points of constellation [0.707+0.707j, -0.707+0.707j, -0.707-0.707j, 0.707-0.707j]
- `symbol_map:` mapping between constellation and symbols [0, 1, 3, 2]
- `soft_decision_precision:` precision of LUT in bits,
- `npwr:` power of noise

Both `constellation_points` and `symbol_map` are used in the coder and decoder.



Variable
ID: symbol_map
Value: 0, 1, 3, 2

Variable
ID: constellation_points
Value: 707m+707... 707m-707mj

Variable
ID: npwr
Value: 1

Variable
ID: soft_decition_precision
Value: 4

Variable
ID: soft_lut
Value: digital.soft_dec_ta...

QT GUI Histogram Sink
Name: QT GUI Plot
Number of Points: 1.024k
Number of Bins: 100
Autoscale: Yes
Accumulate: No
Min x-axis: -1.5
Max x-axis: 1.5

Costas Loop
Loop Bandwidth: 62.8m
Order: 4

Constellation Soft Decoder
Constellation Object: ...=4)>

Constellation Object
ID: qpsk2
Symbol Map: 0, 1, 3, 2
Constellation Points: ...07mj
Rotational Symmetry: 4
Dimensionality: 1
Soft Decisions LUT: soft_lut

Figure 2-22 LUT implementation

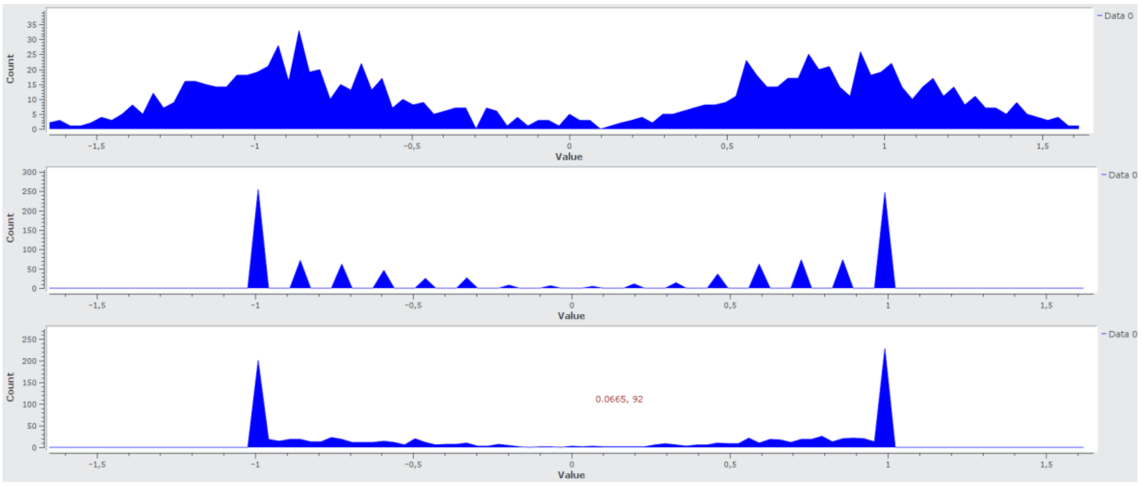


Figure 2-23 Comparison of LLR results for the following methods: direct (top), LUT with 4 bits precision (middle), LUT with 8 bit precision (bottom).

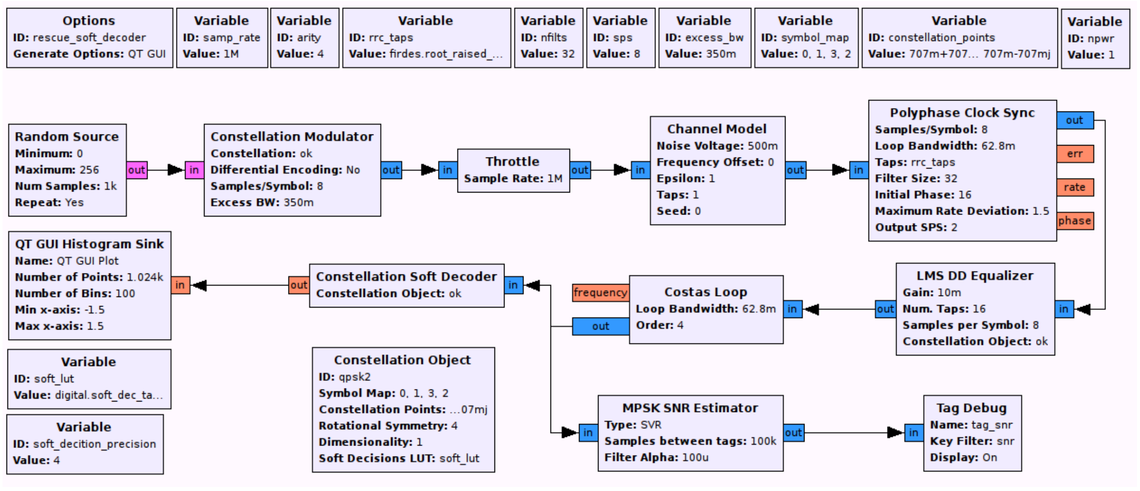
In Figure 2-23, a comparison between the direct method, LUT with 4 bits precision, and LUT with 8 bits precision is presented. One can observe high peak at the value "1" and "-1" for LUT implementation. It is caused by round off results above "1" and below "-1". The second phenomena is related to the short length of the LUT table. For 4 bit precision, the LUT table has 16 position and thus the results are rounded to 16 values. It is clearly visible in the middle image of Figure 2-23, where histogram have only 16 values on the x-axis.

The precision of the simulations is parameterized by means of variable `soft_decision_precision` and should be chosen in the integration stage. During the developing stage, it is better to set a low precision because the calculation of LUT is done after changing any project parameter in GNU Radio and for 8 bit precision takes a few seconds (e.g., after saving any property of any block in the project).

The implementation of LUT should be done by means of Constellation rect. Object instead of the more general Constellation Object (see Figure 2-22). However, during the implementation, an error of GNU Radio,"Constellation rect. Object"" was discovered. The issue as well as software patch has been reported to GNU Radio: http://GNU Radio.org/redmine/issues/784.

2.3.4 SNR Estimation

Estimation of noise power is necessary for symbol remapping. It could be calculated based on SNR which is a more general parameter and used for creating the LUT table (see Section 2.3.3.4). In Figure 2-24, the workflow for SNR estimation for the QPSK modulation is presented. It is based on the MPSK SNR Estimator from GNU Radio.



Options
ID: rescue_soft_decoder
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 1M

Variable
ID: arity
Value: 4

Variable
ID: rrc_taps
Value: firdes.root_raised_...

Variable
ID: nfilts
Value: 32

Variable
ID: sps
Value: 8

Variable
ID: excess_bw
Value: 350m

Variable
ID: symbol_map
Value: 0, 1, 3, 2

Variable
ID: constellation_points
Value: 707m+707... 707m-707mj

Variable
ID: npwr
Value: 1

Random Source
Minimum: 0
Maximum: 256
Num Samples: 1k
Repeat: Yes

Constellation Modulator
Constellation: ok
Differential Encoding: No
Samples/Symbol: 8
Excess BW: 350m

Throttle
Sample Rate: 1M

Channel Model
Noise Voltage: 500m
Frequency Offset: 0
Epsilon: 1
Taps: 1
Seed: 0

Polyphase Clock Sync
Samples/Symbol: 8
Loop Bandwidth: 62.8m
Taps: rrc_taps
Filter Size: 32
Initial Phase: 16
Maximum Rate Deviation: 1.5
Output SPS: 2

QT GUI Histogram Sink
Name: QT GUI Plot
Number of Points: 1.024k
Number of Bins: 100
Min x-axis: -1.5
Max x-axis: 1.5

Constellation Soft Decoder
Constellation Object: ok

Costas Loop
Loop Bandwidth: 62.8m
Order: 4

LMS DD Equalizer
Gain: 10m
Num. Taps: 16
Samples per Symbol: 8
Constellation Object: ok

Variable
ID: soft_lut
Value: digital.soft_dec_ta...

Variable
ID: soft_decition_precision
Value: 4

Constellation Object
ID: qpsk2
Symbol Map: 0, 1, 3, 2
Constellation Points: ...07mj
Rotational Symmetry: 4
Dimensionality: 1
Soft Decisions LUT: soft_lut

MPSK SNR Estimator
Type: SVR
Samples between tags: 100k
Filter Alpha: 100u

Tag Debug
Name: tag_snr
Key Filter: snr
Display: On

Figure 2-24 Implementation of SNR estimation

In the presented workflow, AWGN is added in the Channel Model block as a variance of noise (in Volts). Since the channel model use complex numbers (I/Q samples), it adds noise to the real and imaginary part of the signal. For the QPSK modulation, each symbol has unit energy (constellation: +-0.707+-0.707j), however, noise is added to signal samples and not to symbols. Depending on the settings, there are 2, 4, or 8 samples per symbol.

The MPSK SNR Estimator implements four methods: Simple, Skewness, 2nd and 4th Moment and SVR. All methods were tested and for further analysis the SVR was selected because of its highest precision. The comparison of the measured and calculated SNR values is presented in Figure 2-25. It takes into account the workflow presented in Figure 2-24 and transmissions with 4 samples per symbol.

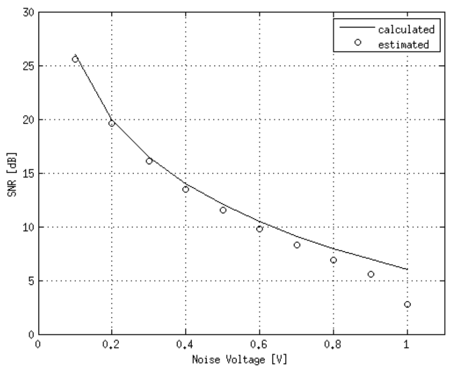


Figure 2-25 Comparison of measured and calculated SNR

One can see that for high power of noise, the estimated SNR is lower than the calculated one. This is caused by the MPSK SNR Estimator which works without a training sequence or other known signal. First, it decodes a symbol and then estimates the error for this symbol (how far the symbol is located form its correct constellation point). This method provides reliable results in low noise environments, however, it fails under high noise levels when the symbol could be decoded incorrectly. For higher accuracy, especially in low SNR environments (i.e., for noisy transmissions), the training sequence has to be used. Therefore, this SNR estimation could not be integrated with the other WP2 developed software. The adopted solution was, in order to avoid transmitting signals that do not carry any information, to use the header of the frame for this purpose. It is guarded by CRC and thus the correctness of its decoding could be the required verification.

2.3.5 Correlation Estimation

As derived in RESCUE Deliverable D2.1.1, the decoding performance at the destination node can be significantly enhanced by jointly decoding several erroneous copies of the same message, which has been received by several relays. Global iterations inside the joint decoder at the destination node exchange LLR information between the multiple relay decoders in order to update unreliable bits from one decoder with reliable bits from another decoder. This process requires reliability information for each erroneous copy of the original message which can be reflected by the bit error rate before encoding at the relays of each message. However, since this information is not available at the destination node, and hardly obtainable at the relays, it needs to be estimated to support the decoding process. Algorithms for the estimation of these error rates have been proposed and analysed under the term *Correlation Estimation* in D2.1.1. In the prototype, two of the proposed algorithms were implemented.

In general, the source correlation estimation is performed in two steps. Initially, a pairwise error probability between pairs of messages is calculated, yielding the pairwise error probabilities q_{ij} between the *i*th and *j*th message. In the second step, the vector of q_{ij} is transformed to actual bit error estimates of each packet. In this process, it is necessary to supply the information which of the messages was known to be error-free, e.g., when it is directly received from the source. A block diagram of the correlation estimator is shown in Figure 2-26.

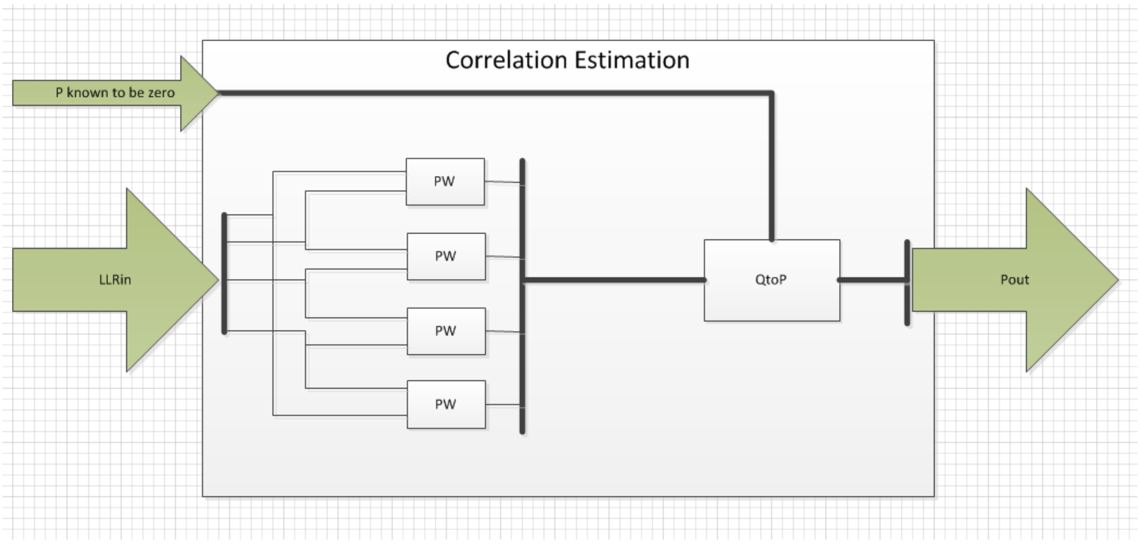


Figure 2-26 Block diagram of source correlation estimation. PW denotes pairwise estimator.

The pairwise estimator implements both the standard algorithm [HZA+13] and the model-based hard-decision algorithm that have been extensively described in RESCUE D2.1.1. The algorithms were chosen for their simplicity and satisfying performance, especially in the case of few relay nodes.

The standard algorithm calculates the pairwise error probabilities according to:

$$q_{ij} = \frac{1}{N} \sum_{n=1}^{N} \frac{\exp(L_i(n)) + \exp\left(L_j(n)\right)}{[1 + \exp(L_i(n))]\left[1 + \exp\left(L_j(n)\right)\right]} \tag{2.19}$$

where $L_i(n)$ is the nth LLR values of the *i*th message and only LLRs that have absolute values above a certain threshold T are considered. For the hard-decision algorithm, the number of LLRs with mismatching signs are counted and finally divided by the overall number of considered LLRs. Furthermore, also for the hard-decision algorithm, a threshold T can be given, limiting the minimum absolute values of LLRs to be considered for correlation estimation.

Finally, transforming the pairwise errors to the errors in the distinct message copies is done as follows. From the system model we can derive

$$q_{ij} = p_i + p_j - 2p_i p_j \tag{2.20}$$

for each pair *i,j*. Here, p_i denotes the bit error rate of message *i*. Hence, for *K* received message copies, with *K* pairwise errors q_{ij}, the system can be solved for the *K* values of $p_i, i = 1, \dots, K$. The corresponding algorithm for the solution of the equation system in given in Figure 2-27 where the required matrices are explicitly given in RESCUE D2.1.1. Note that when there are some p known to be zero, the corresponding rows and columns are to be removed from *A* in order to solve the system.

```
Algorithm 1: p estimator
Input: $\hat{\mathbf{q}}$, $\varepsilon$, Pre-defined maximum iterations $IT_m$
Output: $\hat{\mathbf{p}} \succeq \mathbf{0}$ such that $\hat{\mathbf{p}}$=arg min $||\mathbf{A}\hat{\mathbf{p}} - \hat{\mathbf{q}}||^2$
Initialization: $\hat{\mathbf{p}}^{(0)} = \mathbf{0}$, Calculate $\mathbf{A}$ and $\Delta(0) = ||\mathbf{A}\hat{\mathbf{p}}^{(0)} - \hat{\mathbf{q}}||^2$
for $h = 1$ to $IT_m$ do
    Calculate $\hat{\mathbf{p}}^{(h)}$ by solving (3.6) using *lsqnonneg* algorithm;
    Update $\mathbf{A} = [(\mathbf{I}+\mathbf{J}) - 2 \cdot \mathrm{diag}(\hat{\mathbf{p}}^{(\mathbf{h})}) \cdot \mathbf{J}]$;
    $\Delta(h) = ||\mathbf{A}\hat{\mathbf{p}}^{(h)} - \hat{\mathbf{q}}||^2$;
    if $\Delta(h) \geq \Delta(h-1)$ then
        Stop;
    end
    if $||\hat{\mathbf{p}}^{(h)} - \hat{\mathbf{p}}^{(h-1)}||^2 \leq \varepsilon$ then
        $\hat{\mathbf{p}} = \hat{\mathbf{p}}^{(h)}$;
        Stop;
    end
end
$\hat{\mathbf{p}} = \hat{\mathbf{p}}^{(IT_m)}$;
```

Figure 2-27 Algorithm to transform pairwise error probabilities into message error probabilities

The algorithms are implemented using a general framework that easily allows to extend the pairwise estimation algorithms by supplying the corresponding function object. The main function for correlation estimation is `correlationEstimation()` which accepts the decoded LLRs, the relays that are known to have transmitted error free, and the correlation estimation algorithm to perform. This function then forwards to the corresponding correlation estimation functions in the detailed namespace. In order to create a new correlation estimator, the only thing to be done is to implement the pairwise estimator function based on the LLR inputs. This system makes it very flexible to extend the correlation estimator by more elaborate algorithms, if required.

2.3.6 RESCUE Decoder

Since the received sequences from the source node and relay nodes are highly correlated, a joint decoding process is performed at the destination node with the aim of improving performance. The structure of the joint decoding process is shown in Figure 2-28 and Table 2-6. The joint decoding process includes two parts: local iteration (LI) and global iteration (GI), the details of which are as follows:

1) DeM+ACC^{-1}: The channel state information is estimated by the channel estimator (Section 2.3.5). Thereby, after receiving the signal $\mathbf{y}_k$ from the k-th node, the demapper DeM generates the extrinsic log-likelihood ratio of the modulated symbol sequence, which is input into the ACC decoder ACC^{-1}. A log-maximum *a posteriori* probability algorithm, such as BCJR, is used in ACC^{-1}.

2) D_k: The output LLR sequence from ACC^{-1} is first deinterleaved and then fed into the decoder D_k. The log-MAP algorithm is also performed by D_k. The procedures of DeM+ACC^{-1} and D_k are referred as to LI.

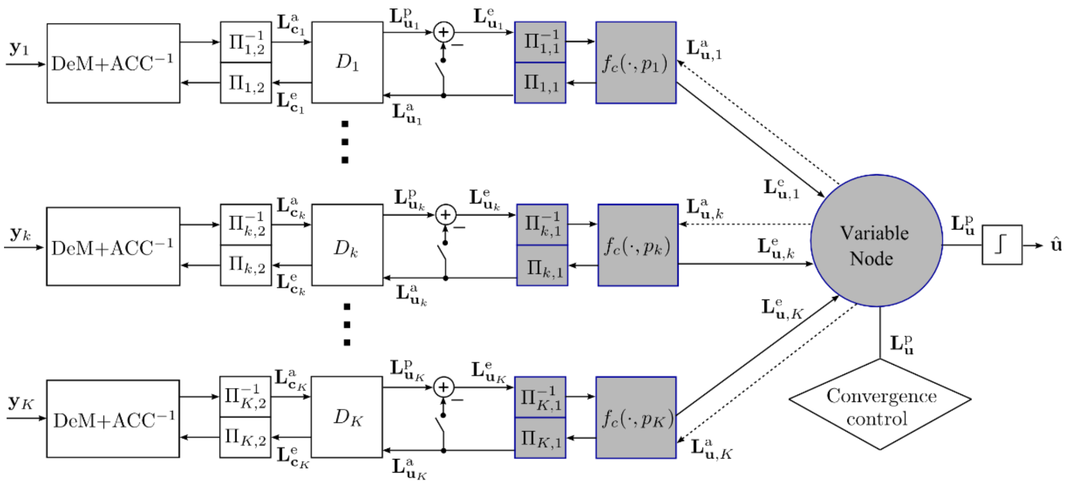
3) GI: The extrinsic LLR output from each LI are updated by the LLR updating function f_c and then added together in the variable node. Afterwards, the output of the variable node is fed back to the

k-th LI after subtracting its own input LLR and updated by the function f_c. The detail of LI and GI can be found in D2.1.1.

4) Convergence control: For saving the computational power of the destination node, we add the convergence control after the GI to remove unnecessary iterations. The mutual information (MI) on the sum LLR $\mathbf{L}_{\mathbf{u}}^{\mathrm{p}}$ is estimated using the averaging method after the second GI. The difference on the MI is calculated by subtracting the value of the last iteration from that of the current iteration. If the difference is smaller than a given threshold, e.g., 10^{-4}, or the number of total iterations reaches the pre-setting value, the iteration process is terminated.

5) Steps 1) – 4) are iteratively executed until the condition of convergence control is not satisfied.

6) Hard decision: The hard decision of $\mathbf{L}_{\mathbf{u}}^{\mathrm{p}}$ is performed to generate the estimates of the source information $\mathbf{u}$.


$\mathbf{y}_1$
DeM+ACC^{-1}
$\Pi_{1,2}^{-1}$
$\Pi_{1,2}$
$\mathbf{L}_{\mathbf{c}_1}^{\mathrm{a}}$
$\mathbf{L}_{\mathbf{c}_1}^{\mathrm{e}}$
D_1
$\mathbf{L}_{\mathbf{u}_1}^{\mathrm{p}}$
$\mathbf{L}_{\mathbf{u}_1}^{\mathrm{e}}$
$\mathbf{L}_{\mathbf{u}_1}^{\mathrm{a}}$
$\Pi_{1,1}^{-1}$
$\Pi_{1,1}$
$f_c(\cdot, p_1)$
$\mathbf{L}_{\mathbf{u},1}^{\mathrm{a}}$
$\mathbf{L}_{\mathbf{u},1}^{\mathrm{e}}$
$\mathbf{y}_k$
D_k
$f_c(\cdot, p_k)$
$\mathbf{L}_{\mathbf{u},k}^{\mathrm{a}}$
$\mathbf{L}_{\mathbf{u},k}^{\mathrm{e}}$
Variable Node
$\mathbf{L}_{\mathbf{u}}^{\mathrm{p}}$
$\hat{\mathbf{u}}$
$\mathbf{y}_K$
D_K
$f_c(\cdot, p_K)$
$\mathbf{L}_{\mathbf{u},K}^{\mathrm{e}}$
$\mathbf{L}_{\mathbf{u},K}^{\mathrm{a}}$
Convergence control

☐ : LI ▨ : GI

Figure 2-28 Structure of the joint decoder

Table 2-6 Implementation of the joint decoder

Item	Function	Input	Output	Remarks
Demapper DeM	`demapper()`	1) feedback LLR (float vector), 2) received signal (float vector), 3) mapping rule (integer vector), 4) mapping symbol (float vector) 5) symbol number (integer) 6) bits per symbol (integer) 7) noise variance (float)	Extrinsic LLR sequence (float vector)	The noise variance is assumed to be known in the demapper. Since the received signal and mapping symbol are complex value, the real and imagination parts are separately input into the function.
ACC^{-1} and D_k	`bcjr()`	1) trellis structure (integer vector) 2) a priori LLR for information bits (float vector)	void	The parameter 6) and 7) are set to static empty vector for ACC^{-1} since they are

		3) a priori LLR for the coded bits with even indices (float vector) 4) a priori LLR for the coded bits with odd indices (float vector) 5) a posteriori LLR for information bits (float vector) 6) a posteriori LLR for the coded bits with even indices (float vector) 7) a posteriori LLR for the coded bits with odd indices (float vector) 8) number of outputs (integer) 9) memory size (integer)		not needed.
Interleaver	`interleave()`	1) LLR sequence (float vector) 2) permutation order (integer vector)	Interleaved LLR sequence (float vector)	---
Deinterleaver	`deinterleave()`	1) LLR sequence (float vector) 2) permutation order (integer vector)	deinterleaved LLR sequence (float vector)	---
fc function	`fc()`	1) LLR sequence (float vector) 2) error probability (float)	fc updated LLR sequence (float vector)	---
Convergence control	`mutual_information()`	LLR sequence (float vector)	mutual information (float)	Set a threshold value, e.g., 10^{-4}, compare the difference between the MI value in the current iteration and that in the last iteration.

2.3.7 RESCUE Soft Deframer

The RESCUE Soft Deframer block extracts the PHY frame from the stream of LLRs outputted by the Soft Demodulator. This is done in the following steps (Figure 2-29):

- search for (correlate) the *access code* in the stream of LLRs coming from the demodulator,
- if access code was found, read and check the *frame length* ,
- if frame length check was passed, read the number of LLRs specified by the frame length, convert the LLRs to a PDU and return it on the output port.

The above described step-by-step procedure is implemented as a state machine with 3-states: SEARCH_SYNC, HAVE_SYNC, HAVE_HEADER, which correspond to the three steps listed above.

```
Algorithm: RESCUE Soft Deframer
Input: stream - stream of LLRs – synchronous (“continuous”)
Output: pdu – RESCUE PHY Frame – asynchronous (“discrete”)
Constants:
#States of the state machine:
SYNC_SEARCH =0, HAVE_SYNC =1, HAVE_FRAME_LENGTH=2
# 64-bit access code
SYNC_VECTOR  = [0xAC,0xDD,0xA4,0xE2,0xF2,0x8C,0x20,0xFC]
# maximum number of wrong bits in the SYNC_VECTOR
SYNC_THRESHOLD = 12
# size of the encoded frame length: 2 int x 2 bytes x 4 (code rate 1/4)
FRAME_LENGTH_SIZE = 64
Initialization:
state = SYNC_SEARCH
frame_length = 0
buffer = []
while stream is not empty:
      llr = read_next_llr(stream)
      buffer.append(llr)
      if state == SYNC_SEARCH:
            # Check if the buffer contains the SYNC_VECTOR
            if count_differences(buffer, SYNC_VECTOR) < SYNC_THRESHOLD:
                  state = HAVE_SYNC
                  buffer = []   #clean the buffer
            else:
                  adjust_buffer_max_size(buffer)
      else if state == HAVE_SYNC:
            if size(buffer) = = FRAME_LENGTH_SIZE:
                  if buffer contains valid frame_length:
                        frame_length = read_frame_length(buffer)
                        state = HAVE_FRAME_LENGTH
                  else:
                        state = SYNC_SEARCH
      else if state == HAVE_FRAME_LENGTH:
            if size(buffer) ==  frame_length:
                  pdu = create_pdu(buffer, frame_length)
                  send(pdu)    # return the PDU to the output port
                  state = SYNC_SEARCH
                  buffer = []     #clean the buffer
```

Figure 2-29 RESCUE Deframer state machine

The RESCUE Frame Format is specified in D3.2. Each PHY frame is preceded by the SFD (Start Frame Delimiter), which plays the role of the access code in the above specified algorithm. The SFD is 64-bits long, the search function allows for a number of erroneous bits in the access code. This threshold is set to 12 by default. Thus the RESCUE PHY frame will be detected, even if 12 out 64 (18.75%) bits are erroneous.

2.3.8 RESCUE PHY Hierarchical Block

The above described blocks are combined into a hierarchical block (Figure 2-30 and Table 2-7), which encapsulates the two flows:

1) Modulation – Converts a message (PPDU) to a complex modulated signal at baseband for USRP Sink
2) Demodulation – Converts complex modulated signal at baseband coming from a USRP Source to a message (PPDU)

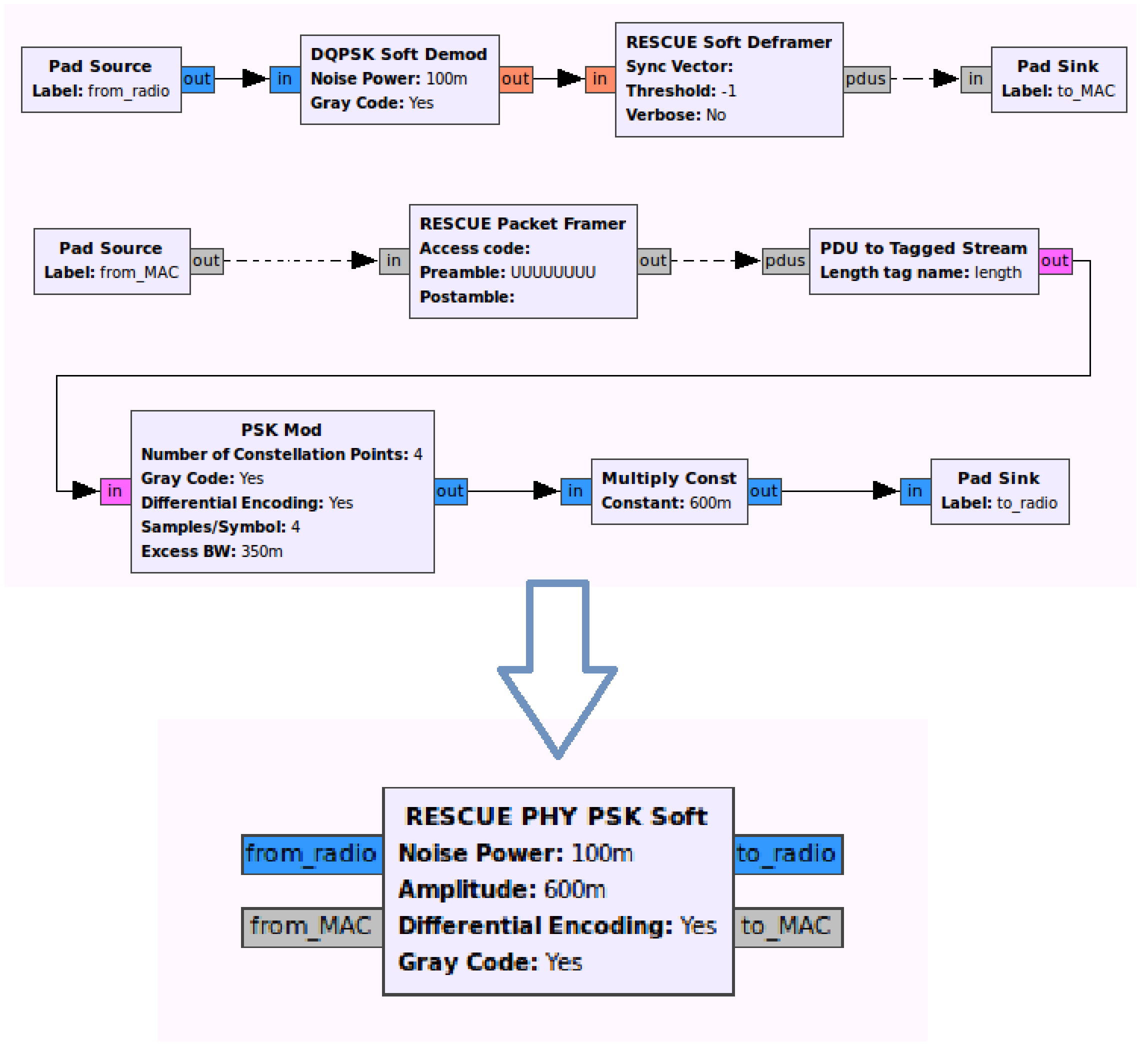


Figure 2-30 Wrapping the modulation and demodulation flows into one reusable block

Table 2-7 Implementation of the RESCUE PHY Hierarchical Block

Block Name	Responsibility	Input	Output
RESCUE PHY PSK Soft (Hierarchical Block)	Modulation – Converts a message (PPDU) to a complex modulated signal at baseband for USRP Sink	from_MAC: PPDU	to_radio: Complex Modulated signal at baseband
	Demodulation – Converts complex modulated signal at baseband coming from a USRP Source to a message (PPDU) Parameters: `noise_power` - specifies the noise variance used in LLR calculation during demodulation	from_radio: Complex Modulated signal at baseband.	to_MAC: PPDU

2.4 RESCUE Relay Node

The operation of the relay node is illustrated on the flow chart in Figure 2-31. The Signal received `from radio` is sent directly to the *RESCUE PHY PSK/16QAM Soft* hierarchical block described above. Exactly the same hierarchical block is also used in the RESCUE destination node. This block performs two main operations: demodulation (performed in *DQPSK/QPSK/16QAM Soft Demod* GRC sub block) and deframing (performed by *RESCUE Soft Deframer*). The de-framer returns a message consisting of soft bits (LLRs). The message contains both the header and payload LLRs. The decoding process, performed in the *RESCUE Decoder* GNU Radio block, starts from decoding the header. If the header decoding process fails (i.e., the checksum fails) the frame is discarded. The information stored in the header is required in the relay node for MAC and/or routing operation. Without it forwarding the frame is not possible, mostly because the relay node has no knowledge about where and how the frame should be transmitted.

However, if the header decoding is successful the decoder begins to decode the payload. In this case it is no longer necessary for the payload decoding process to end successfully. The Lossy Forwarding (LF) concept allows to forward frames decoded unsuccessfully which basically means frames containing errors ("lossy frames"). Therefore, right after decoding, the decoded data is passed to the RESCUE Encoder block, where it is interleaved and re-encoded again. The re-encoded message is passed to *RESCUE PHY PSK/16QAM Soft* where it is framed, modulated and finally send `to radio`.

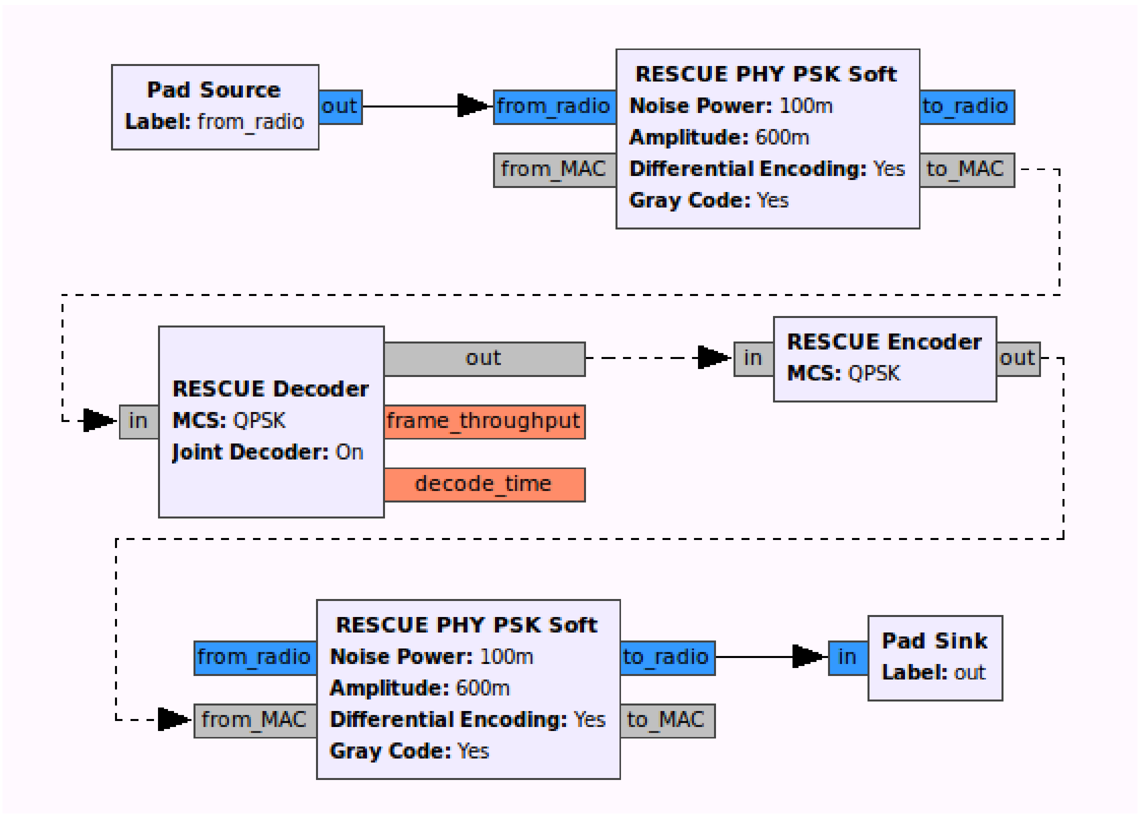


Figure 2-31 Operation of the Relay node

2.4.1 Practical Power Allocation Algorithm

In this subsection, the discussion regarding to the practical power allocation algorithms for implementation can be classified into three different channel configuration cases: For the first case, there is no information exchange among different nodes, even the statistic channel knowledge is not available; For the second case, only the statistic channel knowledge (e.g. the location information of different nodes) can be known by each node; For the third case, there exists a central control unit (e.g. the destination node in our case), which can send feedback bits to different transmission nodes. The next step is to discuss the practical power allocation algorithms for different channel configuration cases in detail.

For the first case, the algorithm can be implemented to any toy scenario (TS) identified in WP1, and unlike the joint power allocation among different transmission nodes, the work here is more likely to let the relay node(s) independently control their transmit power based on the corresponding estimated error probability of the source-to-relay (S-R) link(s). Such strategy strictly sticks to the "links-on-the-fly" concept. In detail, according to Shannon's lossy source-channel separation theorem, the relationship between the relay node's SNR and the error probability of S-R link (i.e. p_e) can be built up. Then, the algorithm here is to adaptively scale the relay node's transmit power with α to formulate the actual transmit power, where α is a scaling factor formulated according to $\mathrm{p_e}$, e.g., $\alpha = 1 - \mathrm{p_e}$.

For the second case, the work is mainly focusing on TS1 and TS2 identified in WP1, and the algorithm is to joint allocate source and relay nodes' transmit power by following certain objectives. In detail, assume that each transmission node has the statistic channel knowledge of all links. Such channel knowledge can be obtained by the off-line training process. Then, the outage probability of decode-and-forward relaying system allowing intra-link errors can be formulated. Here, as we described in D1.2.1, the outage event happens when the source node's transmission rate falls outside the achievable rate region. In this case, the power allocation algorithm can either minimize the outage probability subject to the total power constraint, or minimize the total power constraint subject to the outage probability requirement. Compare with the equal power allocation, the proposed algorithm can offer better performances.

For the third case, the work is mainly focusing on TS2 identified in WP1, and the power allocation algorithm is used to allocate the multiple relay nodes' transmit power based on the corresponding S-R links' error probabilities obtained at the destination node. Consequently, feedback needs to be provided by the destination node to allocate the relay nodes' transmit power. In detail, the destination node can estimate the error probability of S-R links based on the work in D2.1.1. Then, with the estimated error probability, the power scaling factor $\alpha_{\mathrm{i},\forall \mathrm{i}}$ can be formulated at the destination node and then fed back to all the corresponding relay nodes, where $\sum_{\mathrm{i}} \alpha_{\mathrm{i}} = 1$. More detailed explanation about the proposed algorithm design can be found in D2.2.2.

3. Simulations Results

This section covers three topics: (i) the simulation software and models that were used, (ii) the simulation scenarios that were evaluated, (iii) the results and their analysis

3.1 Simulation Software and models

Three simulators have been developed and used in the RESCUE project: RESCUE Coding Algorithms – the RCA simulator, the RESCUE Coder in the GNU Radio simulator and the RESCUE PHY Demo simulator.

3.1.1 RESCUE Coding Algorithms

RCA (RESCUE Coding Algorithms) is a console program written in C++ and implementing the Basic Simulation Model (BSM), for a single link simulation analysis. The RCA BSM model is presented in Figure 3-1.

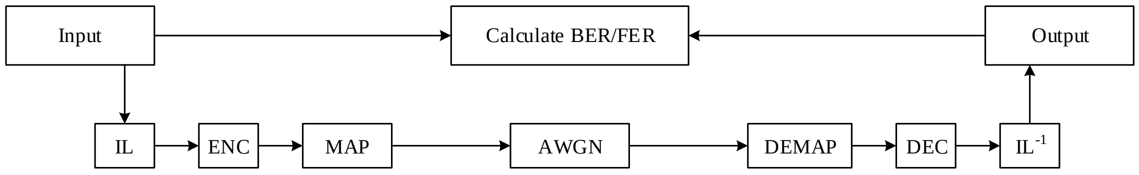


Figure 3-1 Basic simulation model for single link

Where:

- **Input** – is responsible for generating the input bit sequence (frame)
- **IL** – is a random interleaver (different for each bit sequence)
- **ENC** – is the RESCUE Encoder
- **MAP** – is a modulation mapper. RCA enables using BPSK, QPSK or 16QAM modulation
- **AWGN** – Gaussian Noise parametrized by σ^2 (standard deviation of the Gaussian Noise)
- **DEMAP** – is a modulation demapper (BPSK, QPSK, 16QAM)
- **DEC** – is the RESCUE Decoder
- **IL^{-1}** – is a deinterleaver (same as input interleaver but different for each bit sequence)

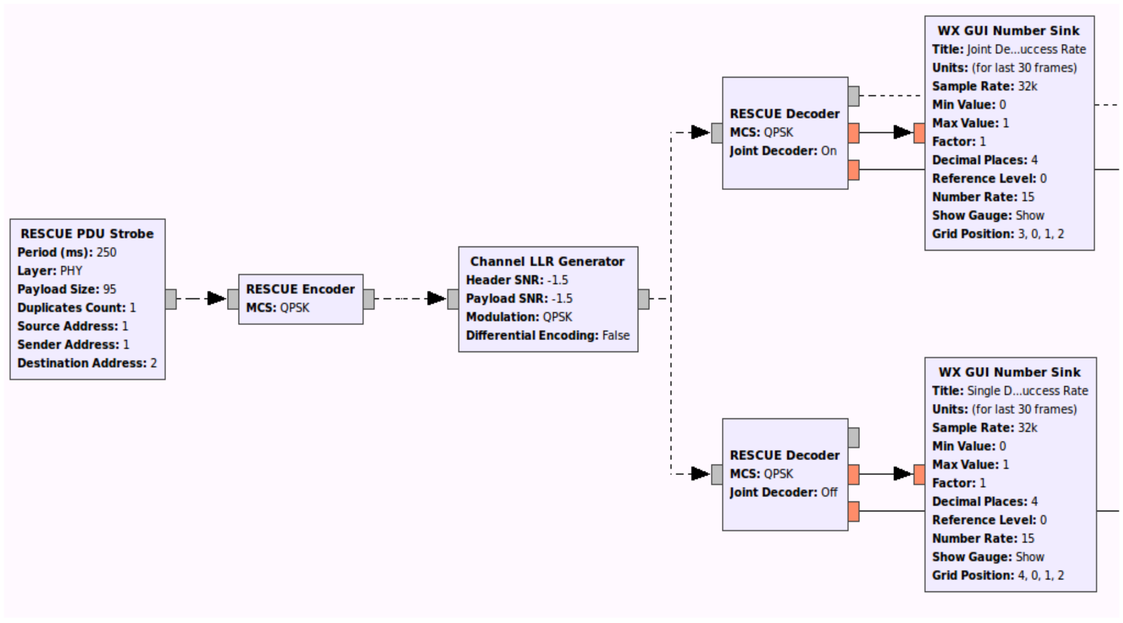
The main goal of the RCA simulator was to generate the reference results in the form of $FER(SNR)$ charts. Basically, the RCA simulator has nothing in common with the GNU Radio environment, but the RCA's implementation of the RESCUE encoding, decoding and interleaving algorithms gave a starting point for the implementation of the GNU Radio blocks. According to this, the RCA’s source code was compiled to a library, which is used in GNU Radio code blocks. The RCA simulating code can therefore be considered an initial phase of the RESCUE implementation process.

3.1.2 RESCUE Coder in GNU Radio

The RESCUE Coder is a GNU Radio flow-graph as shown in the Figure 3-3 below. The presented flow-graph depicts the RESCUE Coding Algorithms, which has been implemented into the GNU Radio environment. It implements the same Basic Simulator Model as RCA using the LLR Channel Generator block. It allows to simulate a toy scenarios using differential and non-differential modulations. It operates on real RESCUE frames, i.e., the PHY PDU, which consists of the PHY Header and PHY Payload. For a single link it gives exactly the same results as RCA. Therefore, the GNU Radio RESCUE Coder implementation can be considered the second phase of the RESCUE implementation process.

The RESCUE PDU Strobe (in Figure 3-3) is a block generating the asynchronous stream of data units which then are forwarded to the RESCUE Encoder block. The whole process of information decoding is based not directly on the received symbols, but on the Log-Likelihood Ratio (LLR). Then, the received stream is split into two parallel streams in order to compare the impact of joint decoder during the decoding process.

The main difference between RCA and the RESCUE Coder simulator is the header processing phase. The RCA simulator does not validate the header checksum and the receiver processes all frames transmitted by the source. Therefore, using RCA it is possible to test and verify the theoretical gain of the combining after decoding technique. In contrast to RCA, the RESCUE Coder simulator validates the header checksum before decoding. If the header is corrupted then the entire frame is discarded. In this case not all transmitted frames are combined after decoding in the destination node. If the header error rate is high, which is common for low SNR values, many frames are discarded and the combining after decoding technique does not provide a visible gain because simply there is nothing to combine in the receiver.


RESCUE PDU Strobe
Period (ms): 250
Layer: PHY
Payload Size: 95
Duplicates Count: 1
Source Address: 1
Sender Address: 1
Destination Address: 2

RESCUE Encoder
MCS: QPSK

Channel LLR Generator
Header SNR: -1.5
Payload SNR: -1.5
Modulation: QPSK
Differential Encoding: False

RESCUE Decoder
MCS: QPSK
Joint Decoder: On

WX GUI Number Sink
Title: Joint De...uccess Rate
Units: (for last 30 frames)
Sample Rate: 32k
Min Value: 0
Max Value: 1
Factor: 1
Decimal Places: 4
Reference Level: 0
Number Rate: 15
Show Gauge: Show
Grid Position: 3, 0, 1, 2

RESCUE Decoder
MCS: QPSK
Joint Decoder: Off

WX GUI Number Sink
Title: Single D...uccess Rate
Units: (for last 30 frames)
Sample Rate: 32k
Min Value: 0
Max Value: 1
Factor: 1
Decimal Places: 4
Reference Level: 0
Number Rate: 15
Show Gauge: Show
Grid Position: 4, 0, 1, 2

Figure 3-3 RESCUE non-waveform simulation model

3.1.3 RESCUE PHY Demo in GNU Radio

The RESCUE PHY Demo flow graph, an extension of the RESCUE Coder, adds the waveform to the basic simulation model. RESCUE PHY Demo implements the Extended Simulation Model (ESM), presented in Figure 3-4, by adding the modulation-demodulation blocks (RESCUE PHY PSK Soft block) and a GNU Radio channel model as illustrated in Figure 3-5.

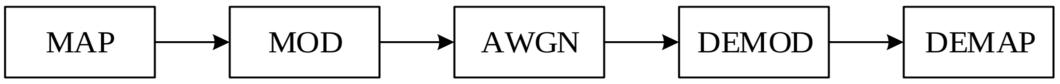


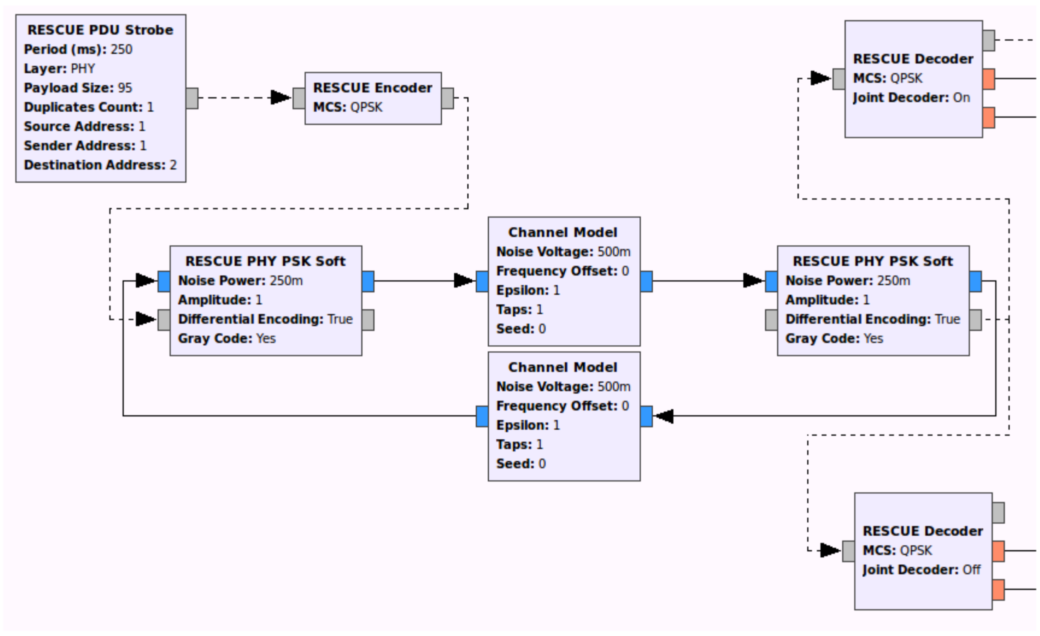
Figure 3-4 ESM – Extended Simulation Model

The RESCUE PHY PSK Soft block represents the physical layer and implements the following radio operations:

- Modulation - responsible for conversion of PPDUs into a complex modulated signal at the baseband, ready to be passed to the USRP sink;
- Demodulation - responsible for conversion of a complex modulated signal received from the USRP Sink into a PPDU;
- Signal shaping.

The RESCUE PHY PSK Soft block is described in detail in Section 2.3.8.

The GNU Radio block flow-graph presented the Figure 3-5 appears to contain a feedback loop between two RESCUE PHY PSK Soft blocks. This is only an implementation related issue, which requires all inputs and outputs not to be left not-connected. In practice the RESCUE PHY PSK Soft is implemented to work in two directions having the inputs: `from MAC` and `from RADIO` and two outputs: `to RADIO` and `to MAC`, respectively. Internal flows connect `from MAC` with `to RADIO` and `from RADIO` with `to MAC` connectors.



RESCUE PDU Strobe
Period (ms): 250
Layer: PHY
Payload Size: 95
Duplicates Count: 1
Source Address: 1
Sender Address: 1
Destination Address: 2

RESCUE Encoder
MCS: QPSK

RESCUE Decoder
MCS: QPSK
Joint Decoder: On

RESCUE PHY PSK Soft
Noise Power: 250m
Amplitude: 1
Differential Encoding: True
Gray Code: Yes

Channel Model
Noise Voltage: 500m
Frequency Offset: 0
Epsilon: 1
Taps: 1
Seed: 0

RESCUE PHY PSK Soft
Noise Power: 250m
Amplitude: 1
Differential Encoding: True
Gray Code: Yes

Channel Model
Noise Voltage: 500m
Frequency Offset: 0
Epsilon: 1
Taps: 1
Seed: 0

RESCUE Decoder
MCS: QPSK
Joint Decoder: Off

Figure 3-5 RESCUE PHY Demo

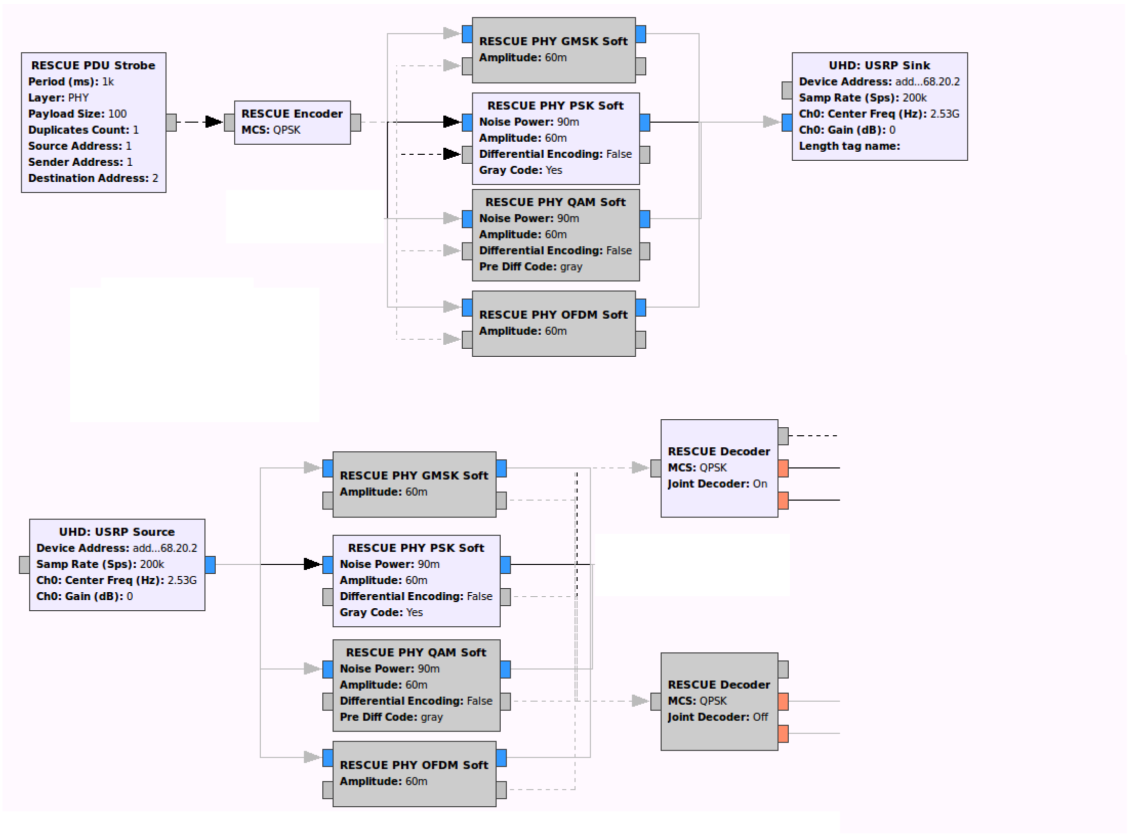
The consequence of adding the waveform to the simulation model is that only differential modulations can produce the same results as the reference experiment results generated by the RCA. Non-differential modulations will work only for high SNR values (even with the USRP and real radio propagation). For lower SNR values, the Phase Lock Loop (Costas Loop) gets out of order and a phase shift can occur in the middle of a received frame. Fortunately, this phase-shift occurs rarely and is not a problem for differential modulation. However, for non-differential modulation it is a problem, since all of the data after the phase shift occurrence is decoded incorrectly.

The example presented in the Figure 3-5 takes only into account the basic GNU Radio Channel Model block (AWGN), however GNU Radio itself supports several different channel models (frequency flat and frequency selective channel models), which can be easily applied into the simulation flow graph.

3.1.4 RESCUE USRP Demo in GNU Radio

RESCUE USRP Demo is the RESCUE Physical Layer Demo executed on the USRP devices. In this flow graph the GNU Radio channel model has been replaced by the USRP Source and Sink blocks. This Demo is the main input to WP4, where it is going to be integrated with the RESCUE MAC. The RESCUE USRP Demo flow-chart is presented in Figure 3-6.

Note, that there are four RESCUE PHY soft modulation/demodulation GNU Radio blocks in the flow graph presented in Figure 3-6. Three of them are greyed out and one is active. It means that RESCUE USRP Demo supports four different modulation types: GMSK, PSK, QAM, and OFDM. In the presented flow graph PSK modulation is active, the rest are disabled.



RESCUE PDU Strobe
Period (ms): 1k
Layer: PHY
Payload Size: 100
Duplicates Count: 1
Source Address: 1
Sender Address: 1
Destination Address: 2

RESCUE Encoder
MCS: QPSK

RESCUE PHY GMSK Soft
Amplitude: 60m

RESCUE PHY PSK Soft
Noise Power: 90m
Amplitude: 60m
Differential Encoding: False
Gray Code: Yes

RESCUE PHY QAM Soft
Noise Power: 90m
Amplitude: 60m
Differential Encoding: False
Pre Diff Code: gray

RESCUE PHY OFDM Soft
Amplitude: 60m

UHD: USRP Sink
Device Address: add...68.20.2
Samp Rate (Sps): 200k
Ch0: Center Freq (Hz): 2.53G
Ch0: Gain (dB): 0
Length tag name:

RESCUE Decoder
MCS: QPSK
Joint Decoder: On

UHD: USRP Source
Device Address: add...68.20.2
Samp Rate (Sps): 200k
Ch0: Center Freq (Hz): 2.53G
Ch0: Gain (dB): 0

RESCUE PHY GMSK Soft
Amplitude: 60m

RESCUE PHY PSK Soft
Noise Power: 90m
Amplitude: 60m
Differential Encoding: False
Gray Code: Yes

RESCUE PHY QAM Soft
Noise Power: 90m
Amplitude: 60m
Differential Encoding: False
Pre Diff Code: gray

RESCUE Decoder
MCS: QPSK
Joint Decoder: Off

RESCUE PHY OFDM Soft
Amplitude: 60m

Figure 3-6 RESCUE USRP over the air demo

3.1.5 Comparison of Simulators

Table 3-1 Comparison of simulators

Features	**RCA**	**RESCUE Coder**	**RESCUE PHY Demo**
Type	C++, Console application	GNU Radio	GNU Radio
Differential Encoding	YES	YES	YES
Mapper/Demapper	YES	YES	YES
Coding Algorithms	YES	YES	YES
Toy Scenario Simulation	NO	YES	YES
PHY Frame Format	NO	YES	YES
Waveform	NO	NO	YES
Fading Channel Model	NO	NO	YES
Packet Framer/Deframer	NO	NO	YES
Connection with USRP	NO	NO	YES
Non-differential modulations: QPSK, 16QAM	YES	YES	NO
Differential modulations: QPSK, 16QAM	YES	YES	YES

Table 3-1 provides a comparison of the features of the three simulators used. In the following, we consider three types of simulation models:

- RCA (AWGN only) for testing TS0-single link,
- GNU Radio without waveform (AWGN only),
- GNU Radio with waveform (AWGN + Frequency selective/flat fading channel model).

3.2 Simulation scenarios

Two simulation scenarios were considered in the simulation analysis.

3.2.1 Toy scenario 0

Toy Scenario 0 (TS0) is a very basic RESCUE scenario where only Source Node (S) and Destination Node (D) exist (Figure 3-7). There is no Relay Node (R) in TS0. In TS0 simulations a single link between the source and destination exists which can be either lossless (when SNR is high) or lossy (for low SNR values). The RESCUE gain can be observed only when the link becomes lossy.

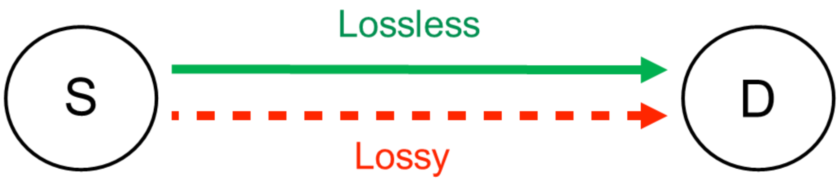


Figure 3-7 RESCUE Toy Scenario 0 – TS0

In TS0, multiple copies of the same frame are sent by the source. The destination receives them and decodes them using the RESCUE joint-decoder. The performance of the joint-decoder is compared to the performance of the single decoder.

TS0 was introduced to the RESCUE project in order to find the SNR range where the RESCUE joint-decoder is able to successfully decode a frame using its multiple copies and the standard single-frame decoder fails. Simulation results obtained with the use of TS0 were used to verify the correctness of coding/decoding algorithms implementation and to find the expected range of SNRs where Toy Scenario 1 simulations should be performed. The knowledge of a single link (TS0) is very important for analyzing and understanding other toy scenarios. The single link analysis clarifies the definition of a lossy-link and is a starting point in the RESCUE Area of Operation estimation process.

3.2.2 Toy scenario 1

Toy Scenario 1 (TS1) is a basic, single-relay model, which consists of a source node (S), a destination node (D) and one relay node (R) (Figure 3-8). There are three relay locations considered in the Task 2.4 simulation analysis:

- Location A, where the relay node is located closer to the source node,
- Location B, where relay is closer to the destination, and
- Location C, in which the three components, source, relay, and destination, keep the same distance from each other, i.e., they form an equilateral triangle.

In all locations A, B and C the coordinates of the source node are $(x = 0, y = 0)$ and the coordinates of the destination node are $(x = 1,\ y = 0)$. The coordinates of the relay node depends on the relay location. In location A the relay coordinates are: $(x = 0.25,\ y = 0.7)$. In location B relay is located at $(x = 0.75,\ y = 0.7)$ and in location C the relay coordinates are $(x = 0.5,\ y = 0.866)$ (Figure 3-8).

Both RESCUE simulators (RESCUE Coder – Section 3.1.2 and RESCUE PHY Demo – Section 3.1.3) take relay coordinates x and y as their input parameters and, based on their values and assuming that the normalized distance between source and destination is set to $d_{sd} = 1$, calculate the relative source-relay d_{sr} and relay-destination d_{rd} distances.

Geometric gain of the source-relay (sr) link with regard to the source-destination (sd) link is defined as:

$$G_{sr} = \left(\frac{d_{sd}}{d_{sr}}\right)^{\alpha} \tag{3.1}$$

It is straightforward to derive the geometric-gain of the relay-destination (rd) link G_{rd} in the same way. More details on geometric gain can be found in the RESCUE Project Deliverable D2.1.1 [RESCUE D2.1.1].

Based on the gemoetric gain and on the source-relay and relay-destination distances the simulators calculate SNRs related to these distances according as

$$\begin{aligned} SNR(d_{sr}) &= SNR(d_{sd} = 1) + \alpha\, 10log_{10}\left(\frac{d_{sd}}{d_{sr}}\right) \\ SNR(d_{rd}) &= SNR(d_{sd} = 1) + \alpha\, 10log_{10}\left(\frac{d_{sd}}{d_{rd}}\right) \end{aligned} \tag{3.2}$$

where the path loss exponent α is assumed to be $\alpha = 3.52$ [YG11].

Simulation results of TS1 present the FER as a function of SNR between source and destination ($SNR(d_{sd} = 1)$ in equation (3.2)).

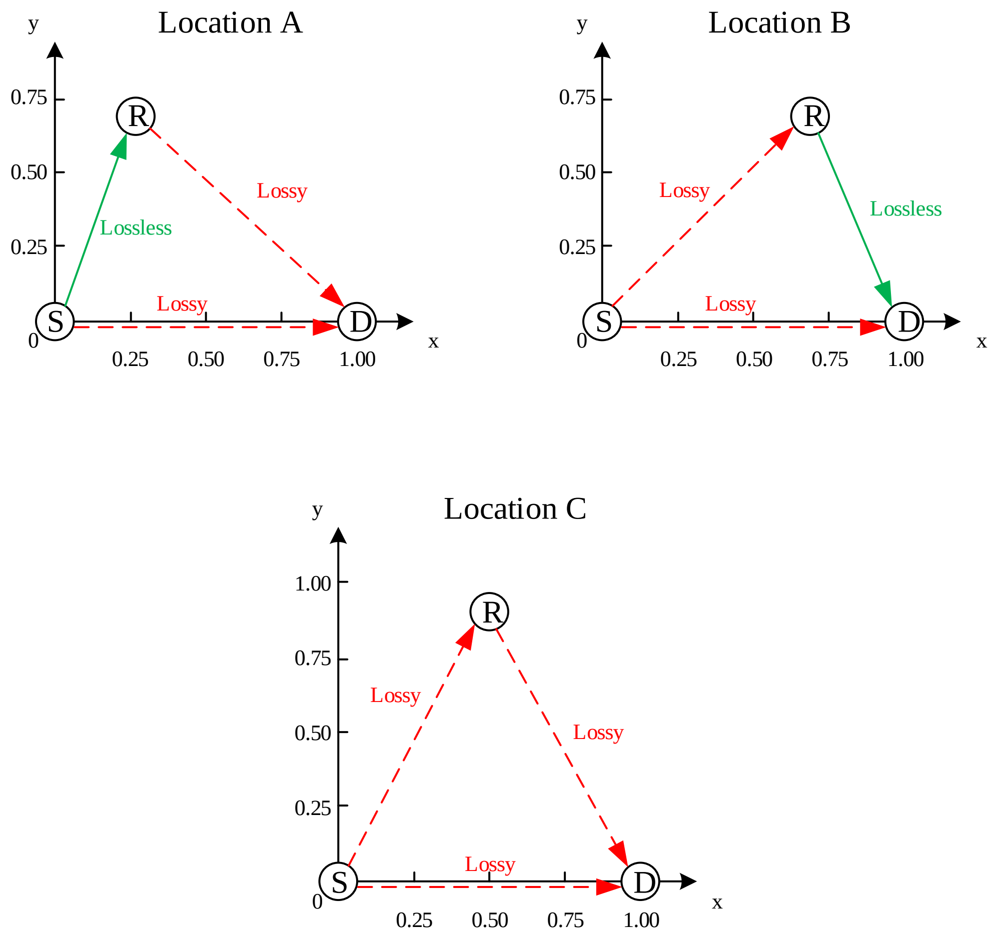


Figure 3-8 Toy Scenario 1 relay node locations

3.3 Simulation parameters and channel models

The common simulation parameters used in all scenarios are as follows:

- Header Length: 248 bits (31 bytes),
- Payload Length: 1200 bits (150 bytes).

Reception of the correct header is extremely important in every communication system. The RESCUE Lossy Forwarding concept allows to forward a frame with erroneous payload. However if the header of the frame received by a relay node is corrupted than the relay cannot forward the frame because it simply does not have the required information where and how the frame should be forwarded. Therefore, the header in the RESCUE frame is protected by a stronger code then the payload. Detailed information about used code rate and header/payload lengths are given in the Table 3-2.

Table 3-2 Common Simulation Parameters used in TS0 and TS1 simulation scenarios

	Overall Code Rate	Uncoded Length	Coded Length
Payload QPSK	1/2	1200	2400
Payload 16QAM	3/4	1200	1600
PHY Header	1/4	248	992

3.3.1 Fading Channel model

The functionality of the RESCUE PHY will be validated considering different types of channel models. While the AWGN channel model is rather simple and widely used for initial performance studies, more realistic models are required to cover the wireless multipath channel behavior of the system/algorithm performance. This has been recognized and considered by different standardization bodies such as ETSI or 3GPP, where tapped delay line (TDL) models or more complex geometry based stochastic channel models (GBSCM, from the WINNER family [RESCUE D4.3]) have been introduced. Since within RESCUE the extended WINNER model for V2V application is not fully implemented at the D2.3 delivery time an interim modelling approach based on the ETSI ITS-G5 channel models [ETSI-ITS-G5] is considered. The approach considered within ETSI ITS-G5 [ETSI-ITS-G5] is based on the standard TDL model, where different sub-scenarios have been considered, e.g., rural LOS, urban approaching LOS, crossing NLOS, highway LOS, highway NLOS. The reported parameters have been derived from channel measurements in real environments. Within WP2 and WP4 the sub-scenario “crossing NLOS” has been selected for the PHY simulation since it is one of the scenarios discussed in WP4 and the WINNER model extension as well as the practical experiments considered. The parameters for the crossing NLOS sub scenario are summarized in Table 3-3.

Table 3-3 TDL parameters of the V2V channel model used in real environment simulation analysis

	Tap1	**Tap2**	**Tap2**	**Tap3**
Power [dB]	0	-3	-5	-10
Delay [ns]	0	267	400	533
Doppler shift [Hz]	0	295	-98	591
Doppler shift used in simulations [Hz]	0	30	30	30
Fading Profile	Rayleigh	Rayleigh	Rayleigh	Rayleigh

Maximum Doppler shifts of taps used in simulations were reduced to values summarized in the “*Doppler shift used in simulations*” row of Table 3-3. The reduction was caused by the low quality synchronization and equalization standard the GNU Radio blocks, which do not support the utilization of a training sequence. When the default channel model Doppler shifts were used, both the receiver and the relay were not able to properly recover the fluctuating frequency and phase. This results in lack of any communication. The only possible solution the overcomes this problem quickly and allows performing the simulation analysis at all was to decrease the frequency and phase. The RESCUE consortium is aware of this problem and will work on the proper solution in the near future.

3.4 Simulation results analysis in AWGN channel model

3.4.1 Simulation results in Toy Scenario 0

Figure 3-9 and Figure 3-10 present BER and FER in function of SNR in an AWGN channel for QPSK and 16QAM modulations respectively. The asymptotic value of BER=0.2 provides information that minimum value of SNR for simulation of "lossy-forwarding" should be SNR ≈ -3 dB for QPSK modulation and SNR ≈ 9.6 dB for 16QAM modulation.

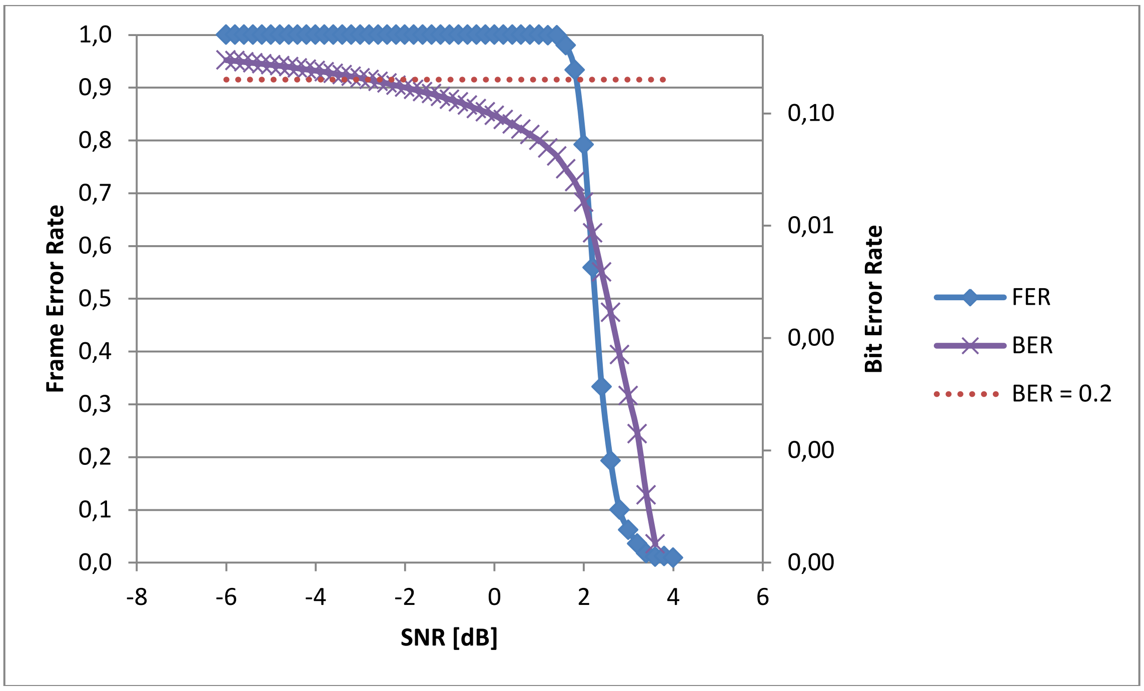


Figure 3-9 RCA Simulation Results: TS0, QPSK, BER & FER vs SNR [dB]

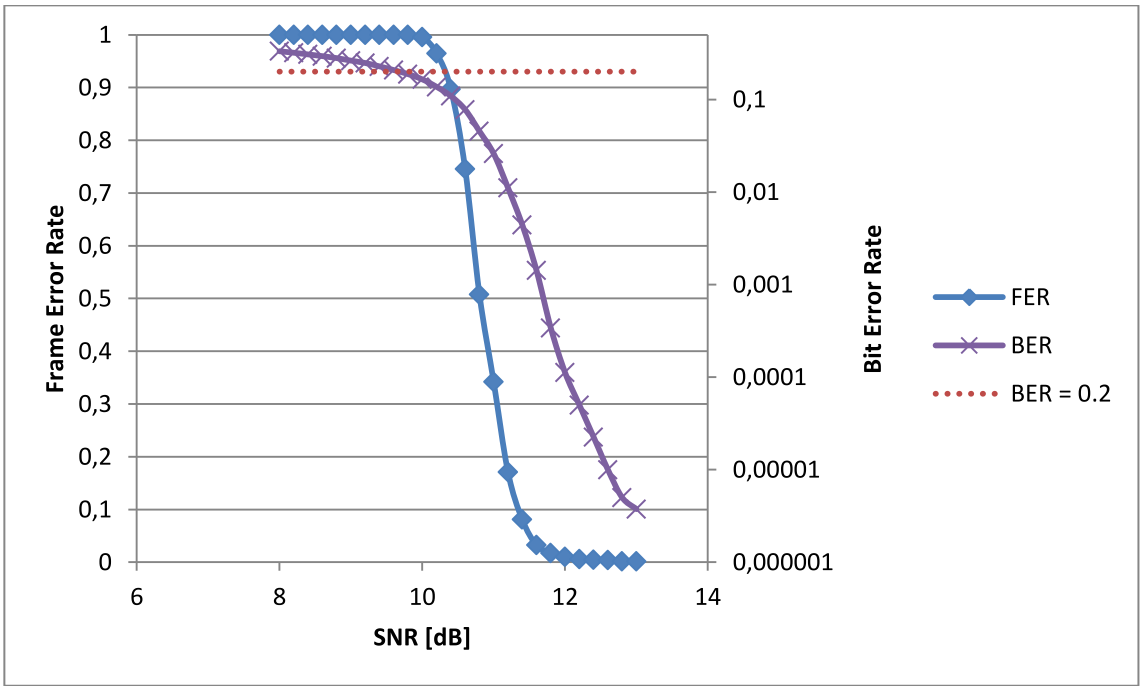


Figure 3-10 RCA Simulation Results: TS0, 16QAM. BER & FER vs SNR [dB]

Simulation results of the combining after decoding FER gain as a function of SNR in the RCA simulator for QPSK in TS0 for an AWGN channel are presented in Figure 3-11. In the RCA simulations we have perfect synchronisation between the transmitter and the receiver. The only errors are due to symbol errors introduced by the AWGN channel. The frame error rate obtained for the case of the single transmission is denoted as FER, frame error rates resulted from combining after decoding of *n* frames are denoted as FER*n*. One can observe that:

- combination of two frames (FER2) results with a processing gain of about 4 dB,
- combination of three frames (FER3) results with a processing gain of about 6 dB,
- combination of four frames (FER4) results with a processing gain of about 7 dB,
- combination of five frames (FER5) results with a processing gain of about 7.5 dB.

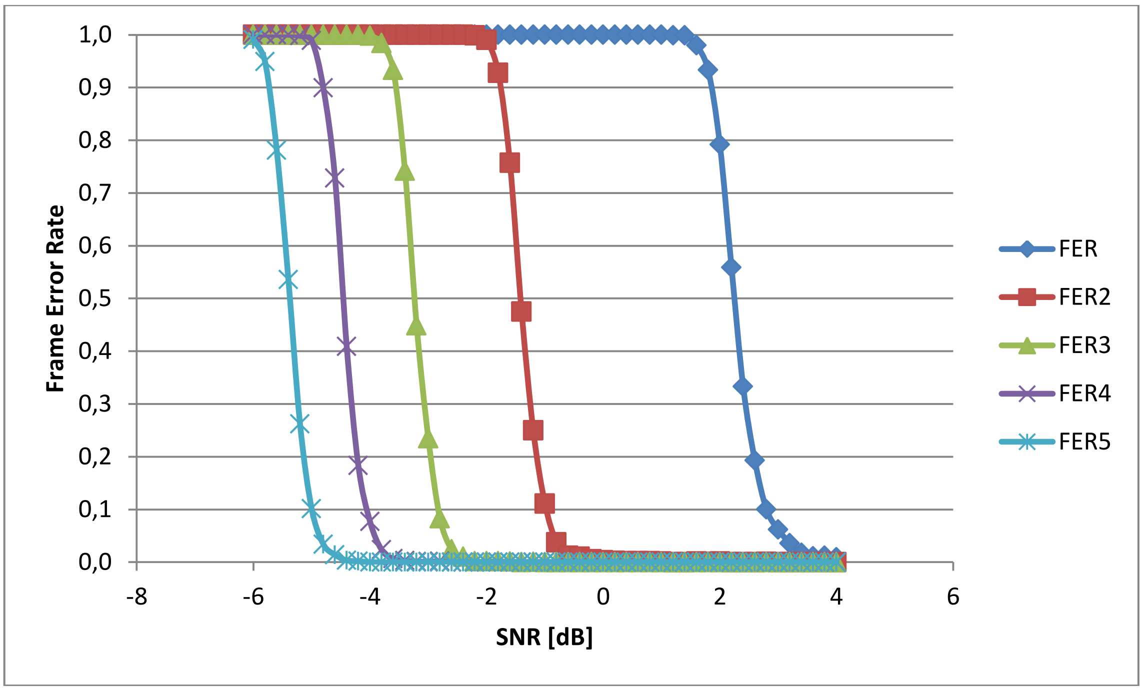


Figure 3-11 RCA Simulation Result:,TS0, QPSK. Combining After Decoding Gain

Based on these results it can be concluded that the growth of the processing gain decreases with an increasing number of combined frames with a geometric progression ($a^n = 4 \times \frac{1}{2^n}$) with an asymptotic value of about 8 dB.

These RCA results provide reference values, including the minimum SNR range, for the GNU Radio simulations.

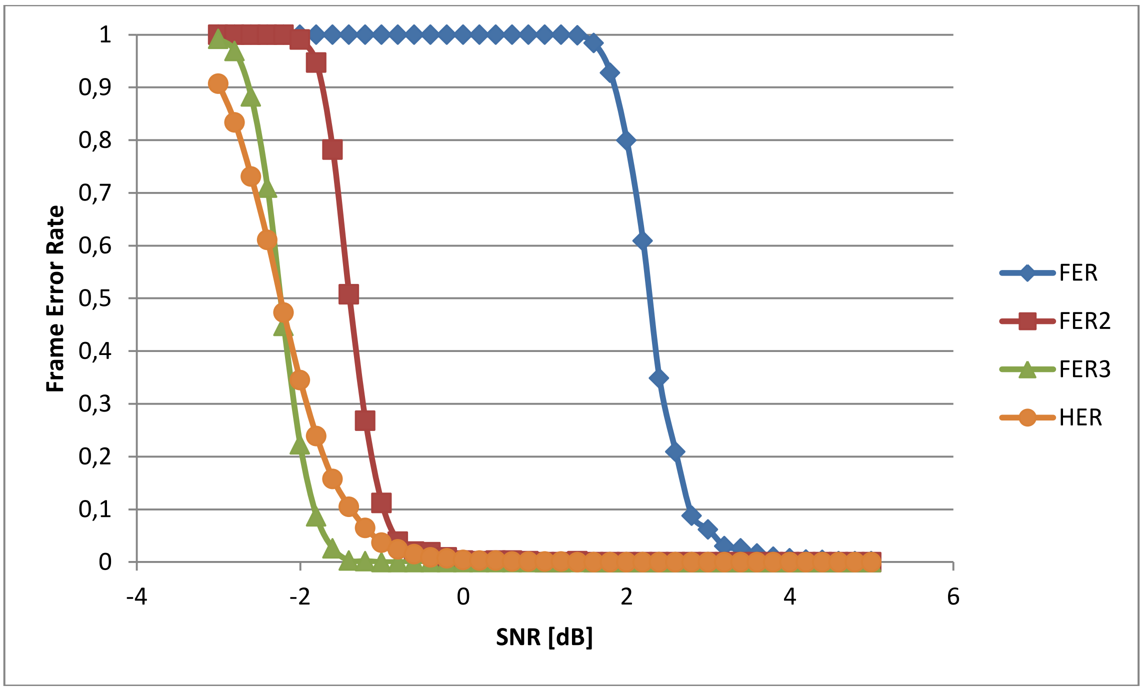


Figure 3-12 Simulation results: TS0, QPSK, non-waveform

Simulation results for the QPSK modulation in TS0 are presented in Figure 3-12. Simulations were performed assuming perfect synchronisation between transmitter and a receiver without waveform.

The frame error rate obtained for the case of a single transmission is denoted as FER, frame error rates resulted from combining after decoding of *n* frames are denoted as FER*n*, while Header Error Rate (HER) represents a corrupted header of a received message with 150 bytes of payload. One can observe that combination of two frames results in a processing gain between 4 and 5dB. A corrupted header results in frame rejection.

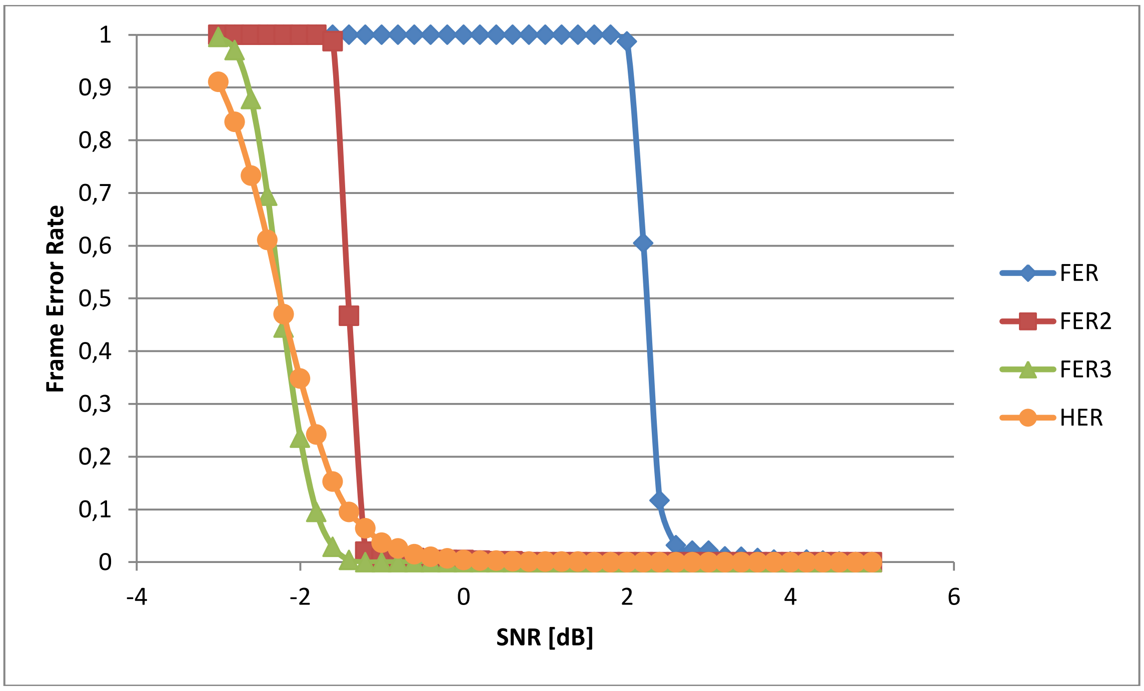


Figure 3-13 Simulation results: TS0, QPSK, non-waveform, payload size 1500 bytes

Simulation results for the QPSK modulation in TS0 are presented in Figure 3-13. Simulations were performed assuming perfect synchronisation between transmitter and a receiver without waveform.

One can observe that combination of two frames results in a processing gain around 4dB, while combination of 3 frames does not give improvement for FER=0. A corrupted header results in frame rejection.

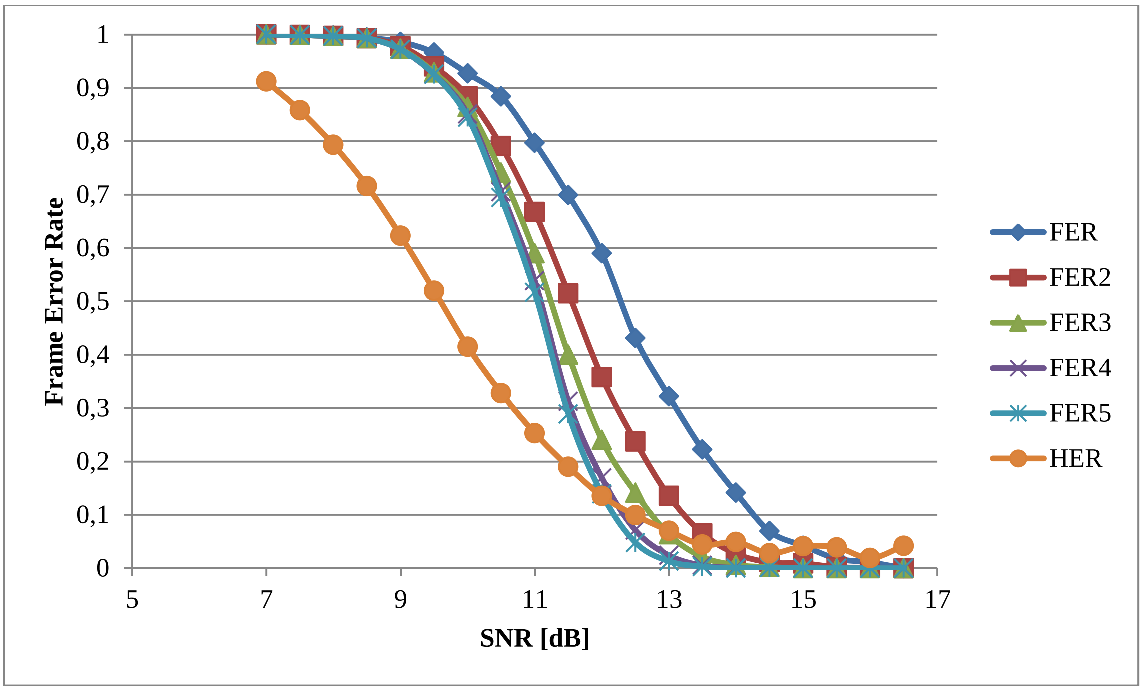


Figure 3-14 Simulation results: TS0, QPSK, waveform

Simulation results for the QPSK modulation in TS0 assuming a waveform between transmitter and receiver are presented in the Figure 3-14. Because of the phase ambiguity problem, the results are 10dB below the expected ones (compare with Figure 3-12). The high required SNR values are the effect of the phase ambiguity problem, which in the case of weak synchronisation, leads to completely wrong symbol decoding, resulting in corruption of large data portions.

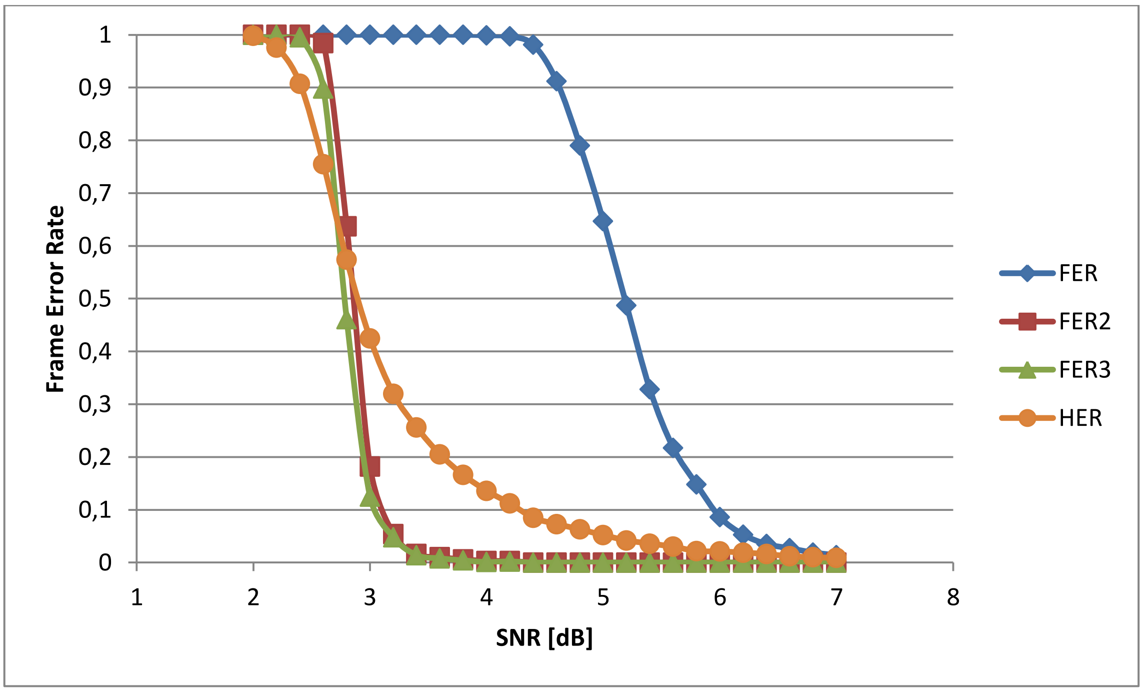


Figure 3-15 Simulation results: TS0, DQPSK, waveform

Simulation results for the DQPSK modulation in TS0 assuming a waveform between transmitter and receiver are presented in Figure 3-15. The frame error rate obtained for the case of a single frame transmission is denoted as FER, frame error rates resulted from combining after decoding of *n* frames (where n=2 or n=3) are denoted as FER*n*, while HER represents a corrupted header of a received message with 150 bytes of payload.

In comparison to the results presented before, the FER=0 case requires higher values of SNR for frames to be transmitted correctly. However, the gain introduced by a combination of *n*-frames (for n=5) at the receiver results with a similar value of the processing gain at level of 3 as it was observed for non-waveform experiments. A corrupted header results in frame rejection.

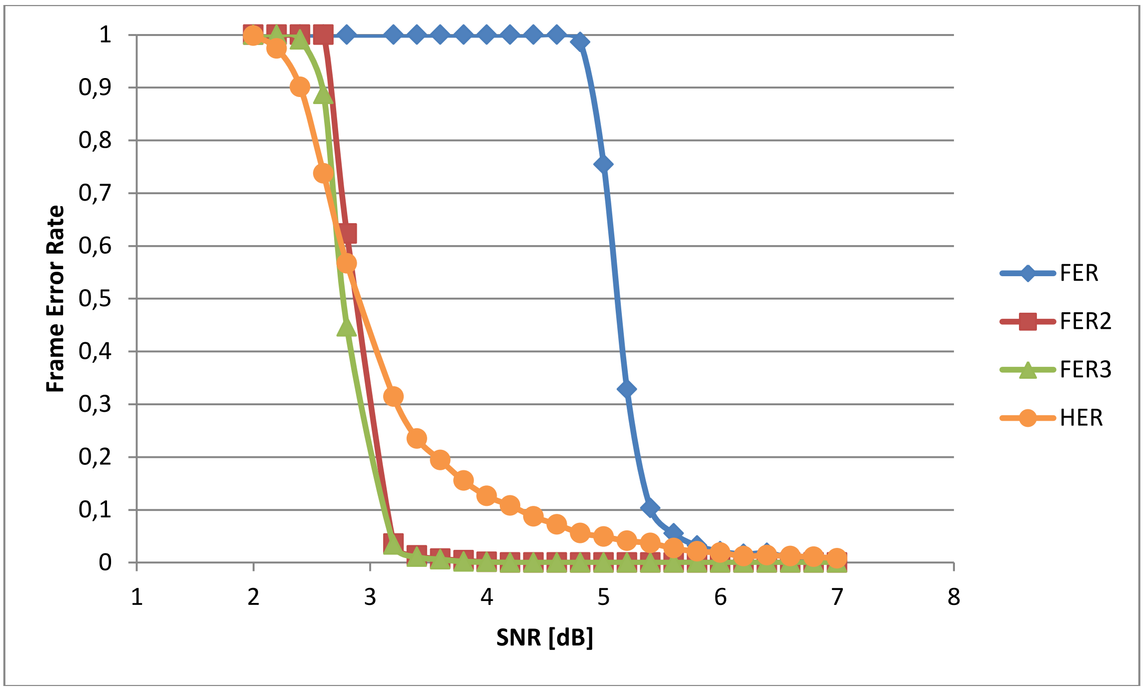


Figure 3-16 Simulation results: TS0, DQPSK, waveform, payload size 1500 bytes

The simulation results for the same case as presented in Figure 3-15, but with a frame payload size of 1500 bytes, are presented in Figure 3-16. As expected the HER in both cases is the same due to the same size of the frame header. The coding algorithm used for frames with 1500 bytes of payload has a very similar performance as the for frames of 150 bytes. .

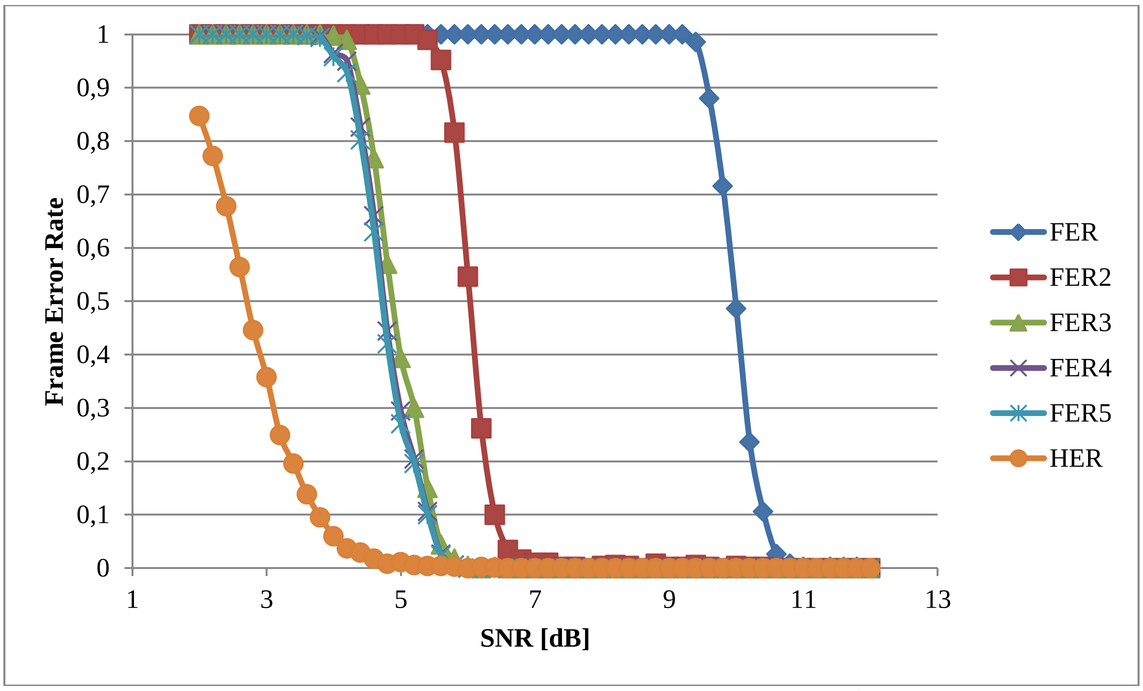


Figure 3-17. Simulation results: TS0, 16QAM, non-waveform

Simulation results for 16QAM in TS0 for the AWGN channel are presented in Figure 3-17. These simulations were performed assuming perfect synchronisation between transmitter and receiver. One can observe that a combination of two frames results with a processing gain of 4 dB. The growth of the processing gain decreases rapidly with an increasing number of combined frames with an asymptotic value of 6 dB.

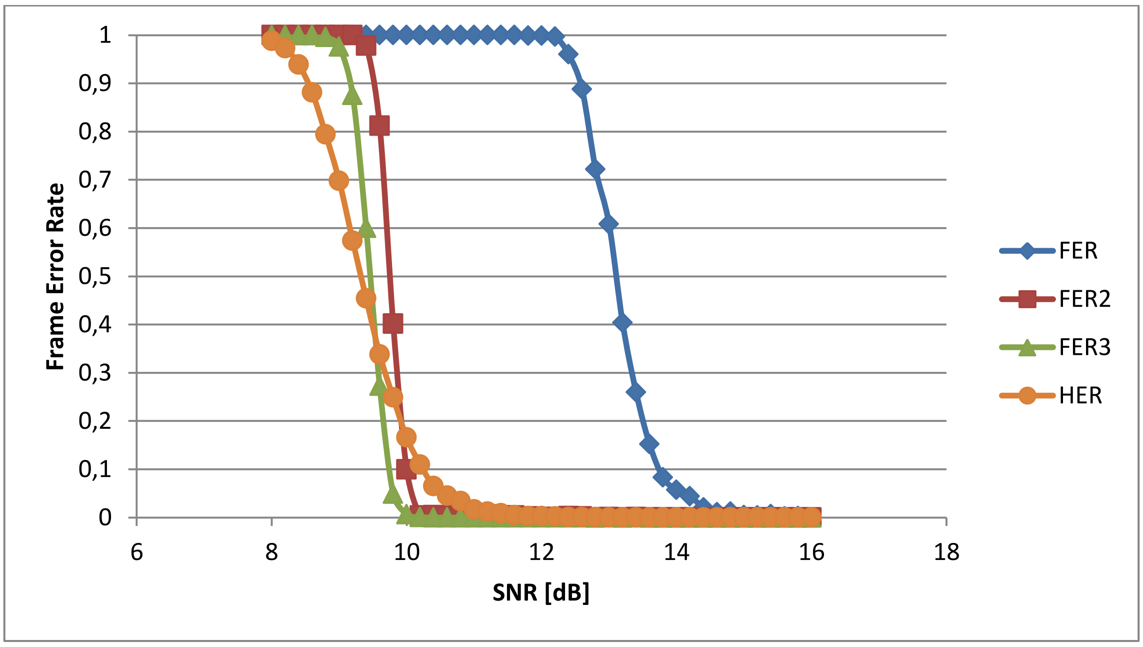


Figure 3-18 Simulation results: TS0, D16QAM, waveform

The results of a more realistic experiment conducted for 16QAM in TS0 for an AWGN channel and real world synchronization algorithms are presented in Figure 3-18. The problem of transmitter-receiver synchronisation including phase ambiguity seriously impacts the performance of the system. A 3 dB performance decrease can be observed for the case of a single transmission. However, the processing gain achieved from processing after decoding of two frames remains at the same level of 4 dB. Contrary to the scenario assuming perfect synchronisation, the processing almost does not grow with the increasing number of combined frames which is most probably bounded by the header error rate.

3.4.2 Simulation results in Toy Scenario 1

Baseline setup

For comparison of the results in TS1, we use two baselines which both use selective decode-and-forward (SDF) with cyclic redundancy check (CRC) as a relaying strategy.

- Baseline 1 (BAS1) – CRC-based SDF at the relay. No joint decoding at the destination; CRC is used to check if the frame is correct or not.
- Baseline 2 (BAS2) – CRC-based SDF at the relay. RESCUE encoding and joint decoding is used.

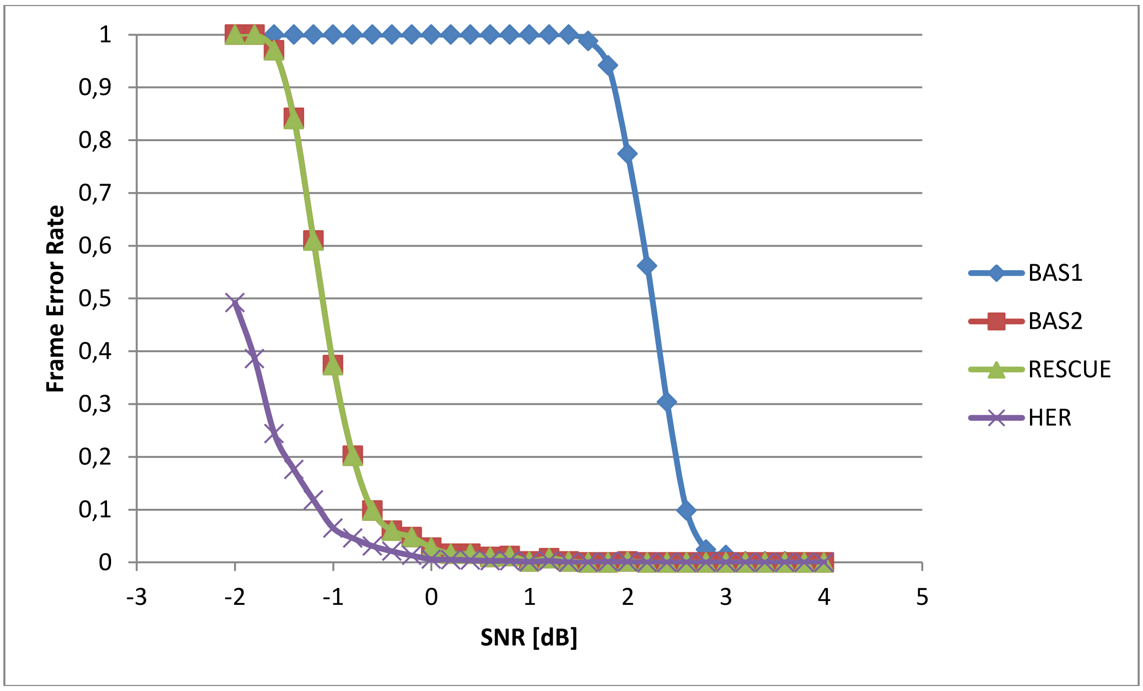


Figure 3-19 Simulation results: TS1, Relay Location A, QPSK, non-waveform

FER vs. SNR for QPSK in TS1 with a relay at location A is depicted in Figure 3-19. We can observe that the RESCUE result is almost the same as in TS0. Furthermore, we observe that in this scenario, Lossy Forwarding is equal to CRC-based SDF. We can justify this by the following remarks on the case of location A:

- The distance between the source and the relay is smaller than the distance between the source and the destination.
- The distance between the relay and the destination is larger than the distance between the source and the destination.

If the SNR of the source-destination link is increased, the relay can decode before the destination. On the other hand, until the sufficiently large SNR value for the relay to correctly decode a frame have been reached, the joint decoded data from the source and to the destination is not error free. The result is exactly what is expected from the information theoretical analysis of the relay channel; it suggests using decode-and-forward (DF) when reliable decoding at the relay node is available [KGG05].

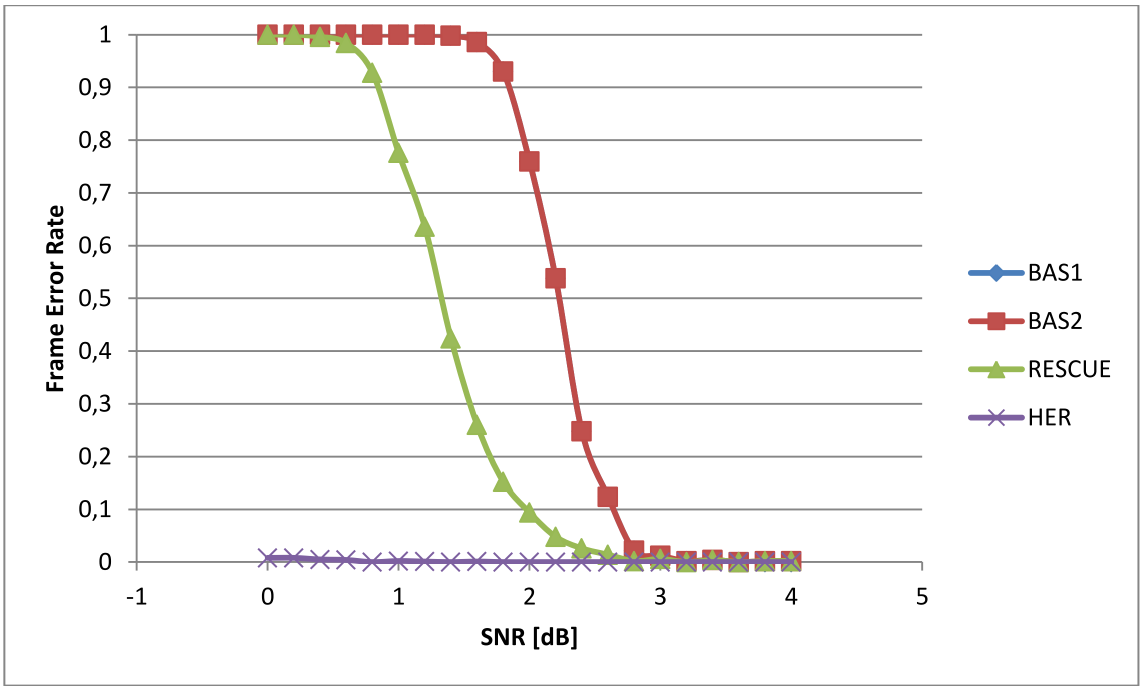


Figure 3-20 Simulation results: TS1, Relay Location B, QPSK, non-waveform

FER vs. SNR for QPSK in TS1 with a relay at location B is depicted in Figure 3-20. Similarly to the case of location A, we can make the following remarks on location B.

- The distance between the source and the relay is larger than the distance between the source and the destination.
- The distance between the relay and the destination is smaller than the distance between the source and the destination.

With an SNR value sufficiently large for the destination to decode error-free, the relay is still unable to decode without errors. Unlike SDF, RESCUE can use this erroneous information. Because of the errors after decoding at the relay, SDF never forwards the data and therefore joint decoding cannot be performed in BAS2 which is therefore equal to BAS1. The gain in RESCUE w.r.t. the baselines is roughly 1 dB.

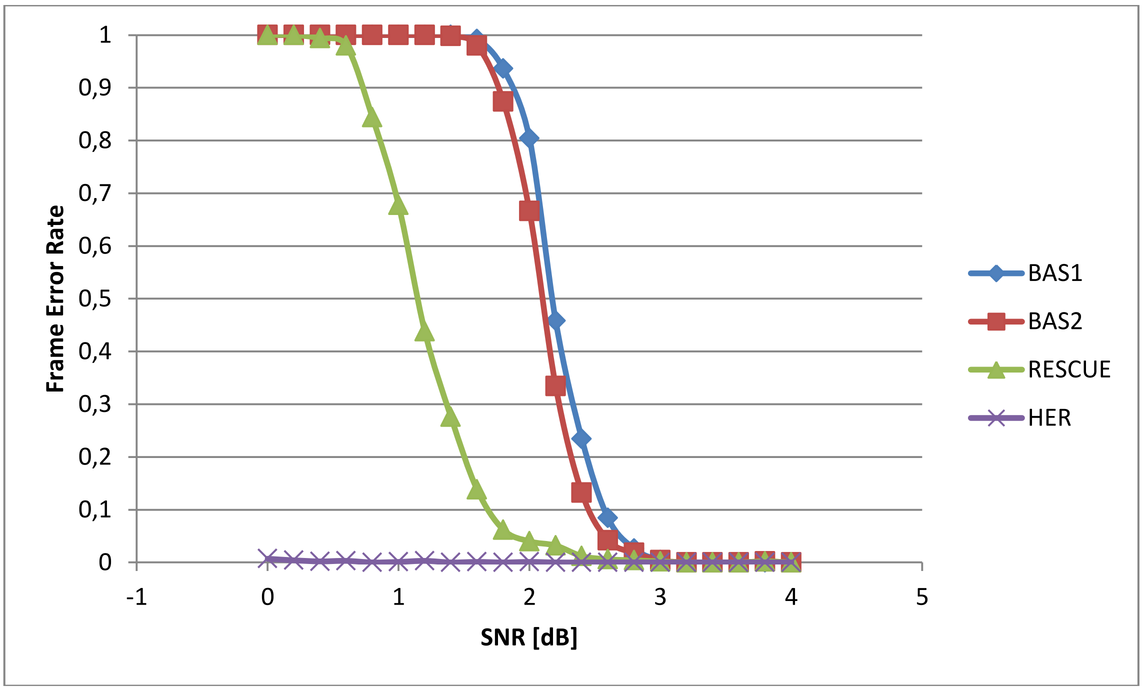


Figure 3-21 Simulation results: TS1, Relay Location C, QPSK, non-waveform

FER vs. SNR for QPSK in TS1 with a relay at location C is depicted in Figure 3-21. The results are similar to the case of location B, except that there is a small gain in BAS2 in comparison to BAS1. It means that the relay can decode and forward a small number of frames for which the joint decoding can be applied in BAS2.

The FER vs. SNR curves for DQPSK in TS1 with a waveform are depicted in Figure 3-22, Figure 3-23 and Figure 3-24 for relays at locations A, B, and C, respectively. The results are similar to the case of QPSK without waveform, however, the difference between the curves are smaller.

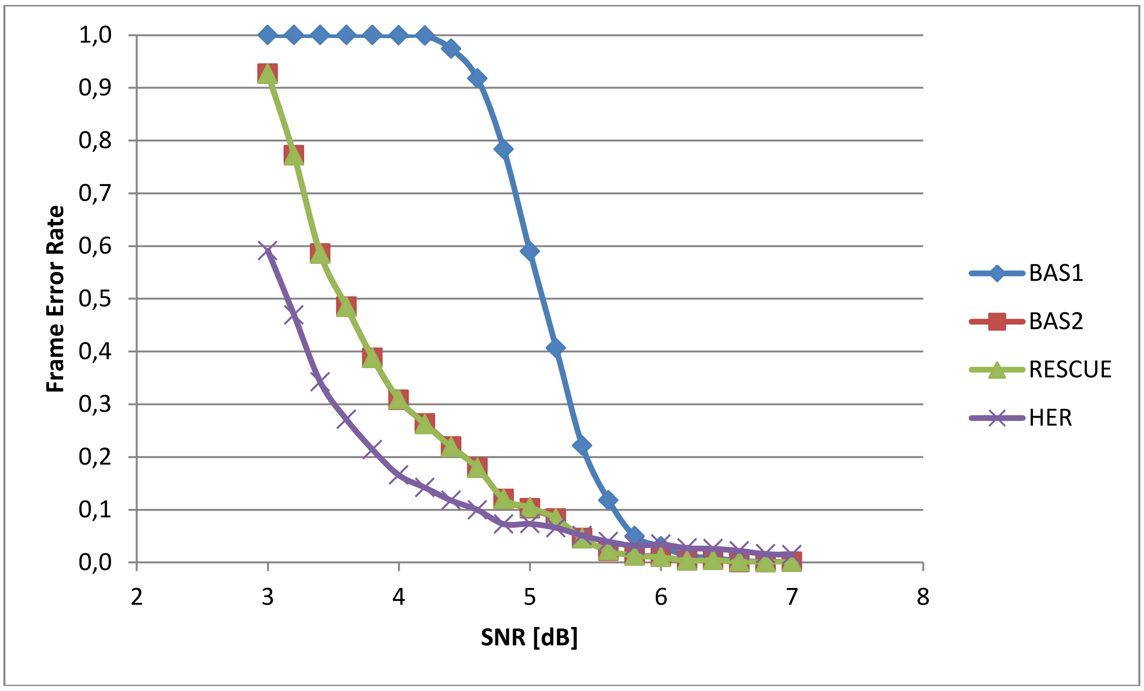


Figure 3-22 Simulation results: TS1, Relay Location A, DQPSK, waveform

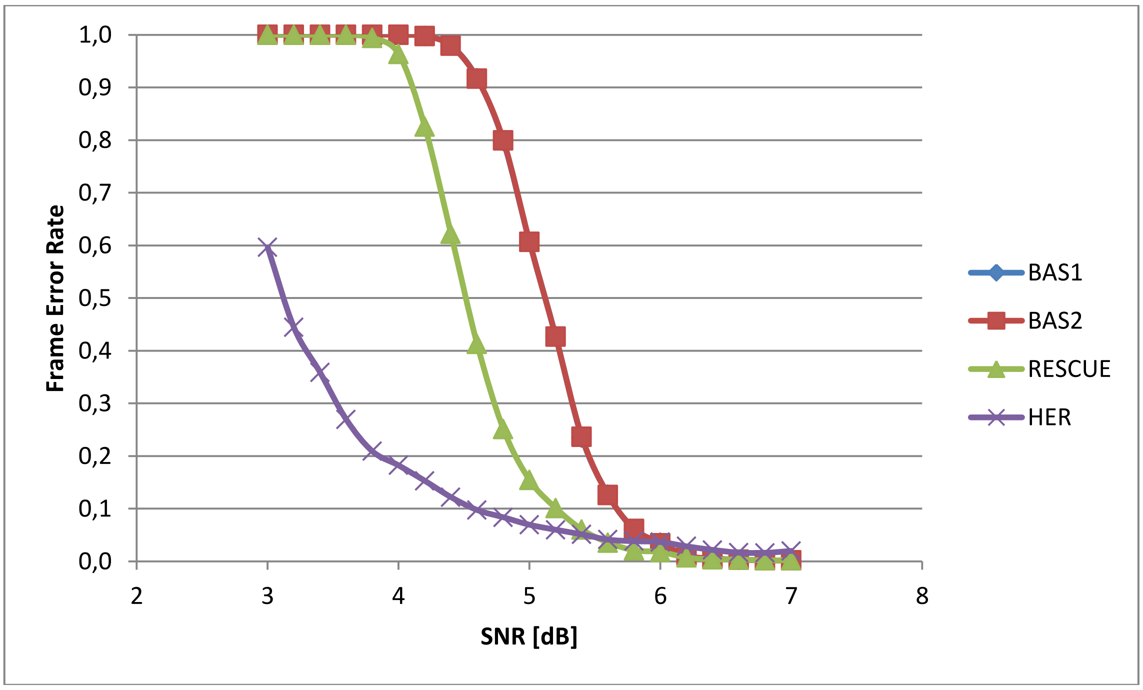


Figure 3-23 Simulation results: TS1, Relay Location B, DQPSK, waveform

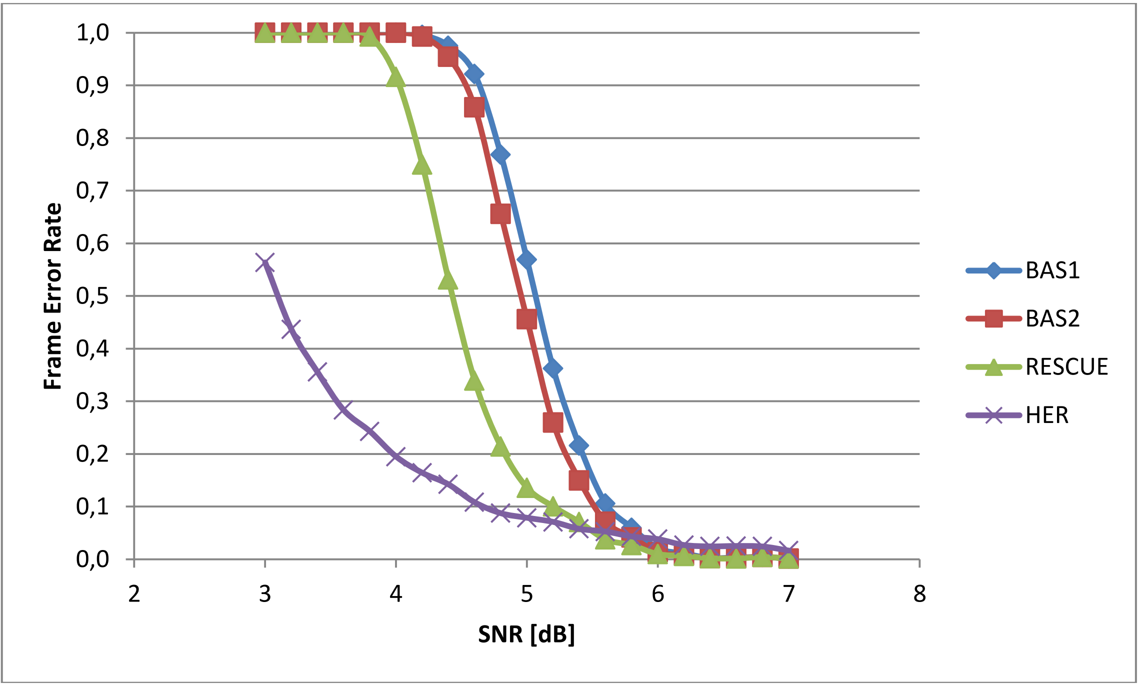


Figure 3-24 Simulation results: TS1, Relay Location C, DQPSK, waveform

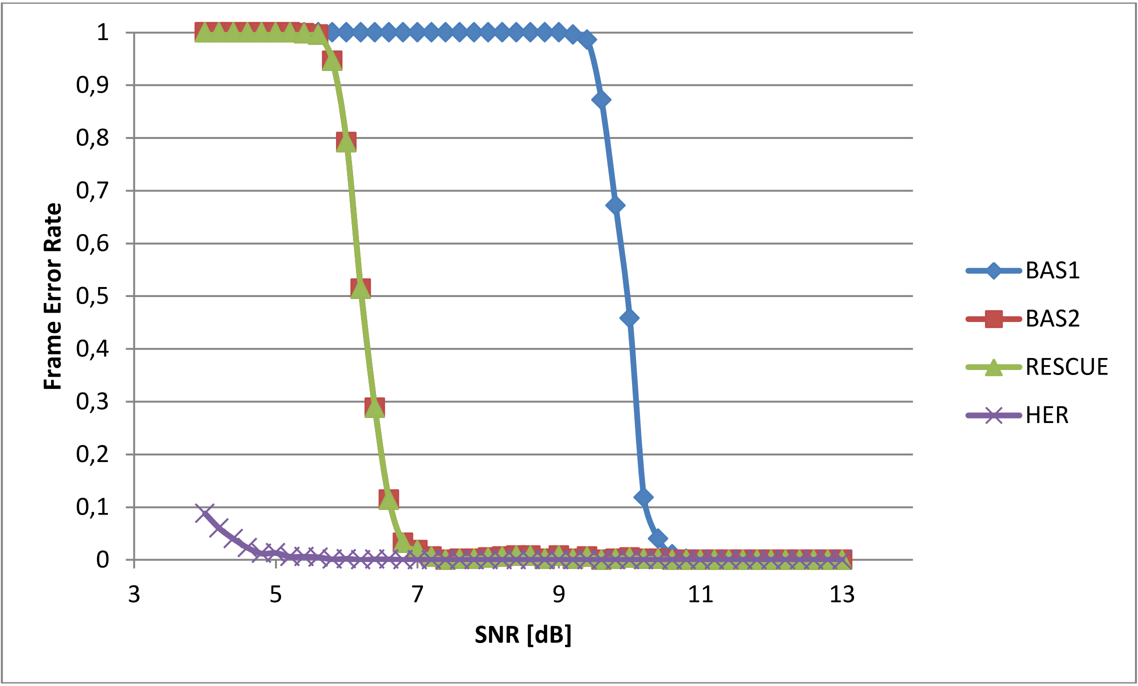


Figure 3-25 Simulation results: TS1, Relay Location A, 16QAM, non-waveform

FER vs. SNR for 16QAM in TS1 with a relay location A is depicted in Figure 3-25. Here we can make the same conclusion as in the case of QPSK; SDF performs equally with Lossy Forwarding.

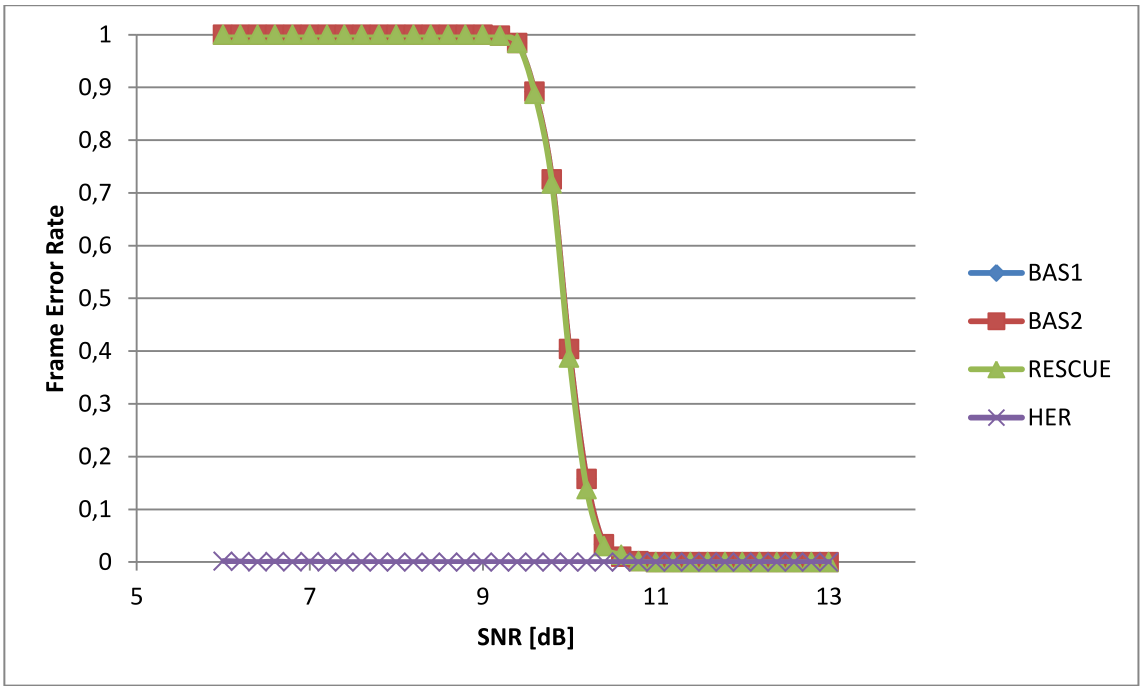


Figure 3-26 Simulation results: TS1, Relay Location B, 16QAM, non-waveform

FER vs. SNR for 16QAM in TS1 with a relay at location B is depicted in Figure 3-26. The results are equal for all relaying and decoding strategies. This can be explained by the fact that the decoder applied to 16QAM has a very steep turbo cliff. This results in a high BER (larger than 0.2) of the forwarded frame in the range of 9 dB to 10 dB (see Figure 3-10). This is not useful anymore to improve the joint decoder performance. At the point where the relay can correctly decode, the destination can also decode error free, and there is no need for a joint decoder.

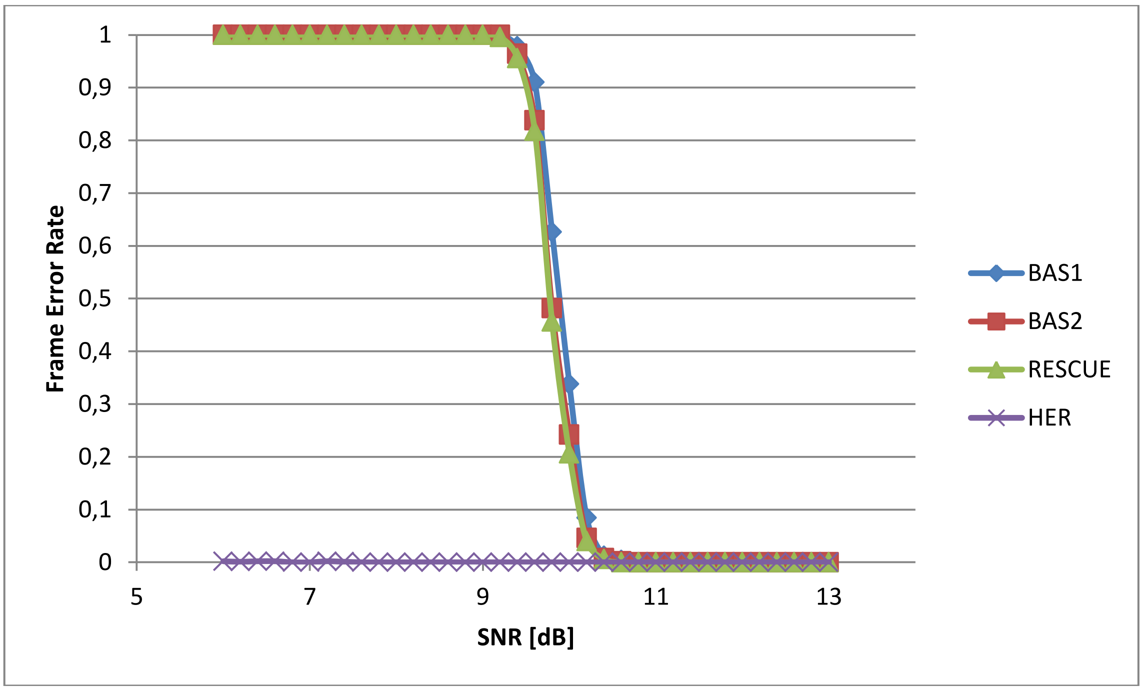


Figure 3-27 Simulation results: TS1, Relay Location C, 16QAM, non-waveform

FER vs. SNR for 16QAM in TS1 with a relay at location C is depicted in Figure 3-27. The results are similar to the case of location B, except that there is a small gain in BAS2 and RECUE in comparison to BAS1. It means that the relay can decode and forward a small number of frames for which the joint decoding can be applied in BAS2 and RESCUE.

3.4.3 Geometrical gain analysis for RESCUE area

This section addresses the geometrical gain (relay and destination locations) for Toy Scenario 1 to show in which scenarios the RESCUE approach can give the greatest benefits. We analyse these potential RESCUE scenarios based on the simulation results of the single link case (Toy Scenario 0) in AWGN channel models. The value of FER (equivalent to the outage probability) and BER are used to indicate the threshold for the lossy-link and no-link region. As shown in Figure 3-28 the S-R and S-D links are lossy, which indicate that the SNR is not high enough to support reliable transmission. Note that the terminology "lossy" here is not the same as that in Lossy Forwarding. Lossy forwarding indicates that errors are introduced before encoding for transmission and the erroneous information sequences are forwarded to the next node in the network. For analyzing the geometrical gain, the SNR is assumed only to be influenced by the distance between nodes. This analysis clearly shows the relay and destination location which constitute a RESCUE scenario and therefore provide important practical guidance to the simulations and analysis in other WPs.

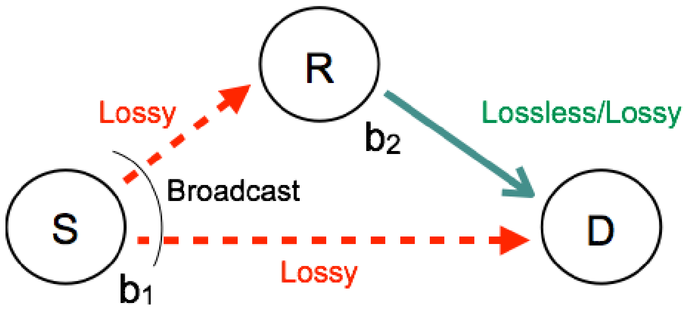


Figure 3-28 RESCUE toy scenario 1

Since the frame is considered erroneous if there is at least one bit that could not be decoded correctly, the information in the erroneous frame still contains useful information for joint-decoding. Therefore, BER=0.2 is defined as a no-link threshold, i.e., if the BER of the link is above this threshold, the link is regarded as there is no transmission between the nodes. Conversely, FER=0.01 is defined as a threshold for lossy-link, i.e., if the FER of the link is higher than this threshold, the link is regarded as lossy, and cannot support reliable transmissions. However, The RESCUE joint-decoder is able to successfully decode the frame by using its multiple copies in lossy-link region.

From the simulation results of Toy Scenario 0 in AWGN channel models, Figure 3-10, we can see that SNR=-3 dB is the threshold for BER=0.2 and SNR=3.8 dB is the threshold for FER=0.01. The SNR interval between the two thresholds is $3.8\,dB - (-3dB) = 6.8dB$. Since the SNR depends on the transmit power and noise power, we set the lossy-link threshold SNR=3.8 dB to be reached when the relative distance d_1 between nodes is one. Then the relative distance d_n for the no-link threshold SNR=-3 dB can be calculated by the geometric gain

$$G = \left(\frac{d_n}{d_1}\right)^{\alpha} \tag{3.3}$$

Then $d_n = 1.56$, given that the path loss exponent is set at $\alpha = 3.52$.

For illustration purposes, we plot the lossy-link region as well as the no-link region for relay and destination nodes onto a Cartesian coordinate system as shown in Figure 3-29, where the source node is located at the origin. If the relay or destination node moves into the lossy-link region (solid circle,$d_1 = 1$), the source-destination link or the source-relay link becomes lossy, respectively. If the relay or destination node is located within the no-link region (dotted circle,$d_n = 1.56$), the source-destination link or the source-relay link can be considered as non-existent. The RESCUE joint-decoder could not decode the frame anymore by using its multiple copies.

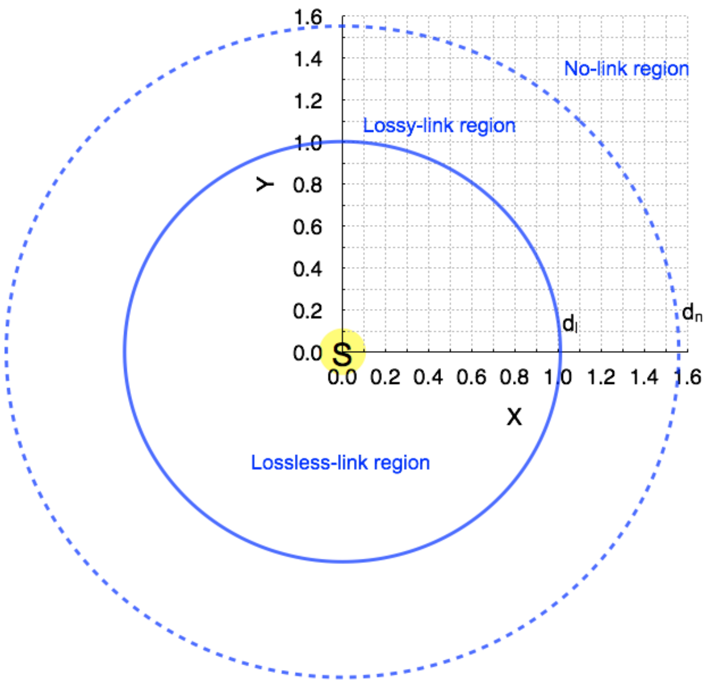


Figure 3-29 Lossy-link region (FER > 0.01 and BER≤0.2) and no-link (BER>0.2) regions for R and D.

The relay and destination locations for RESCUE and non-RESCUE cases are depicted in Figure 3-30. The RESCUE scheme is based on having the transmission errors happen before encoding due to the lossy links. Therefore, the RESCUE case is that the Source-Destination and the Source-Relay links are lossy simultaneously while the Relay-Destination link exists. The solid circle separates the lossless-link region within which lossless transmission can be guaranteed, and the lossy-link region. As shown in Figure 3-30, the scenario when both the Relay and Destination nodes are located inside the lossy-link region can be considered as a RESCUE case. Either the relay or the destination node located outside the lossy-link region is not the RESCUE case. Note that it may be an unrealistic case when the source-relay distance is longer than the source-destination distance (pink node case). However if the source-relay link experiences milder fading than source-destination link, it is possible to have a relay further than destination to source. Moreover, the relay and destination nodes should be located not too far away from each other to keep the relay-destination link exist.

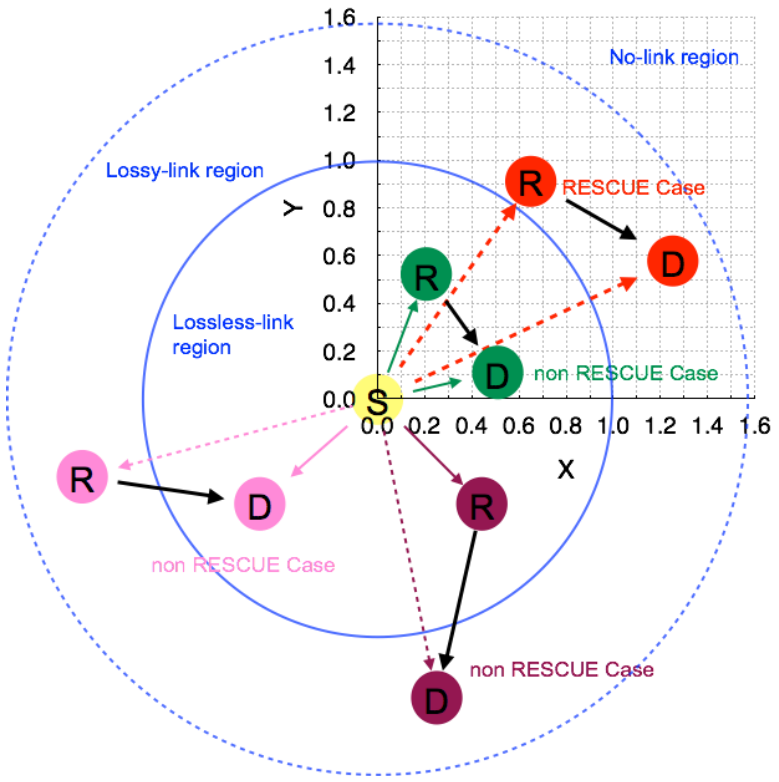


Figure 3-30 Relay and Destination locations for RESCUE and non-RESCUE cases

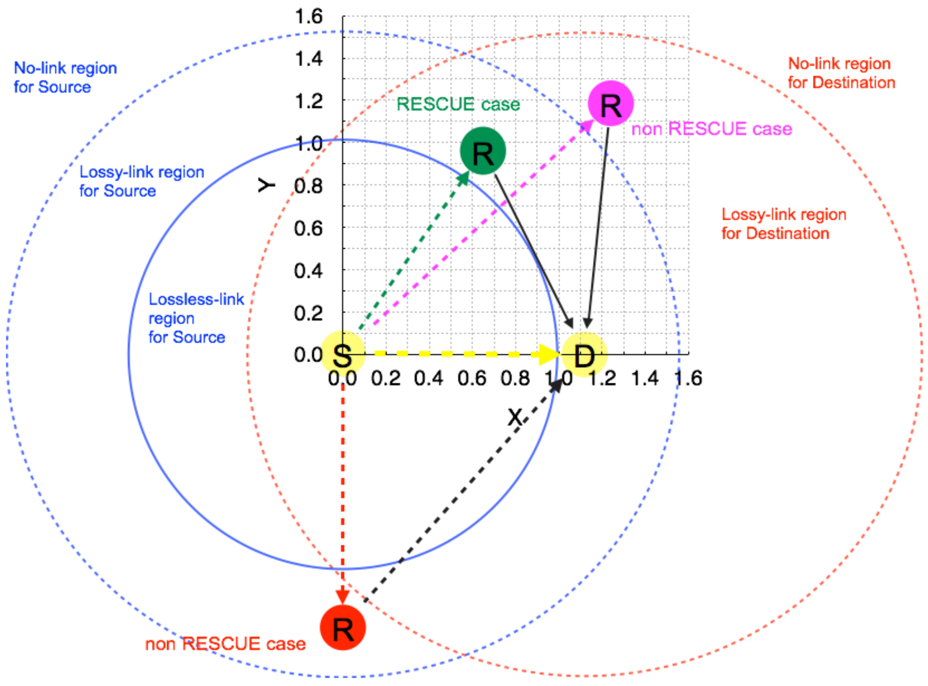

No-link region for Source
Lossy-link region for Source
Lossless-link region for Source
No-link region for Destination
Lossy-link region for Destination
RESCUE case
non RESCUE case
non RESCUE case
S
R
D
X
Y
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6

Figure 3-31 Relay locations where the RESCUE scheme can provide benefits.

In Figure 3-31, the destination is assumed to at the position within the lossy-link region for the source node. Therefore the source-destination link is lossy and cannot support reliable transmissions with standard single-frame decoder. If the system utilizes the relay located also within the lossy-link region, i.e., the green relay node, then we have the RESCUE case. The decoding errors occurring at the relay are due to lossy transmission on the source-relay link. However, with the RESCUE lossy forwarding scheme, the relay re-encodes the data sequences containing errors and forwards it to the destination. Note that the position of the green relay node is also inside the lossy-link region for destination node to ensure the relay-destination link exists. If the relay node is located at a position within the no-link region for source node, i.e., the pink relay node, there can consider no transmission link between source and relay nodes. Therefore, this scenario is not considered in RESCUE. Similarly, if the relay node is located at a position within the no-link region for the destination node, i.e., the red relay node, there will be no transmission link between relay and destination nodes. Therefore, the system becomes a non-RESCUE case, which is also not a scenario where the RESCUE scheme can bring benefits.

3.5 Simulation results analysis in real environment channel models

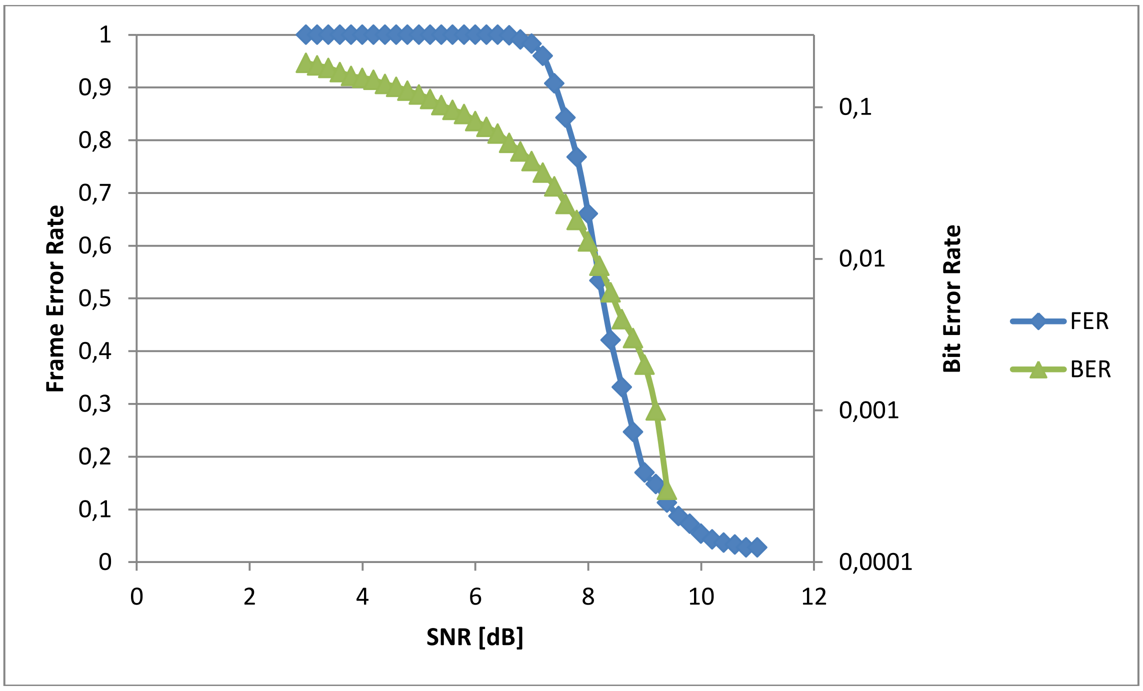


Figure 3-32 Simulation Results: TS0, DQPSK with LMS DD Equalizer

FER and BER for DQPSK with LMS DD equalizer in the presence of frequency selective fading is plotted in Figure 3-32. It is worth of noticing that the decay of the BER curve is milder and therefore, the gains in RESCUE over BAS2 is expected to be larger than in AWGN channel.

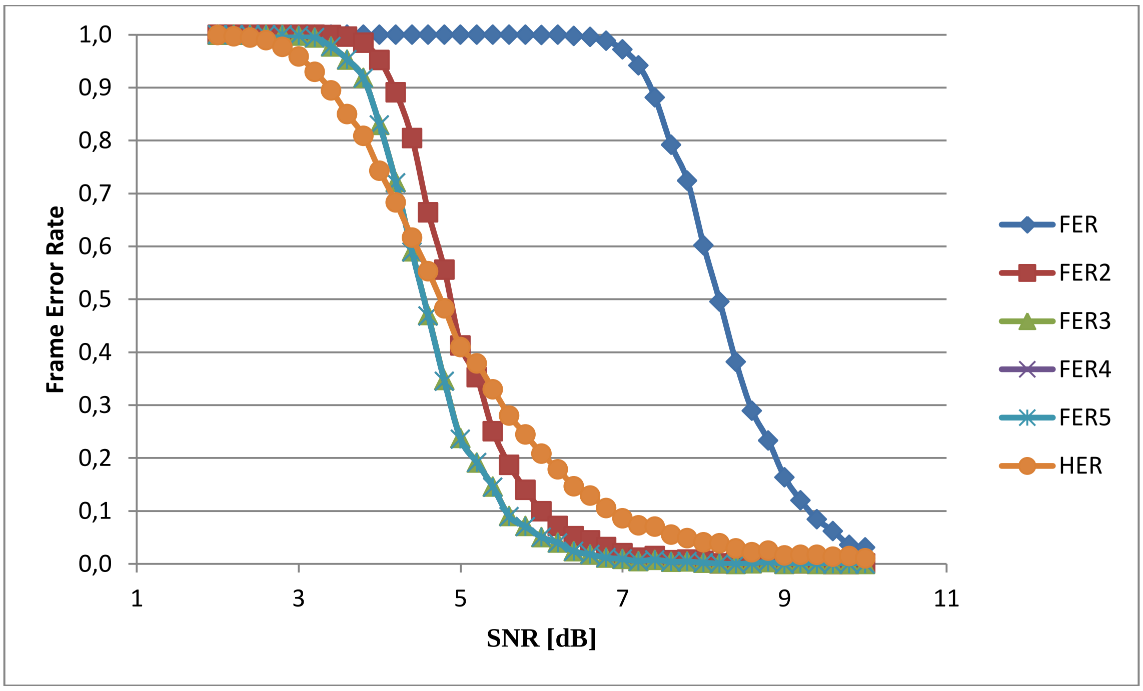


Figure 3-33 Simulation results: TS0, DQPSK, with LMS DD equalizer

FER vs SNR for DQPSK is plotted in Figure 3-33. It can be seen that the required SNR can be decreased by sending more copies as long as the header can be received correctly.

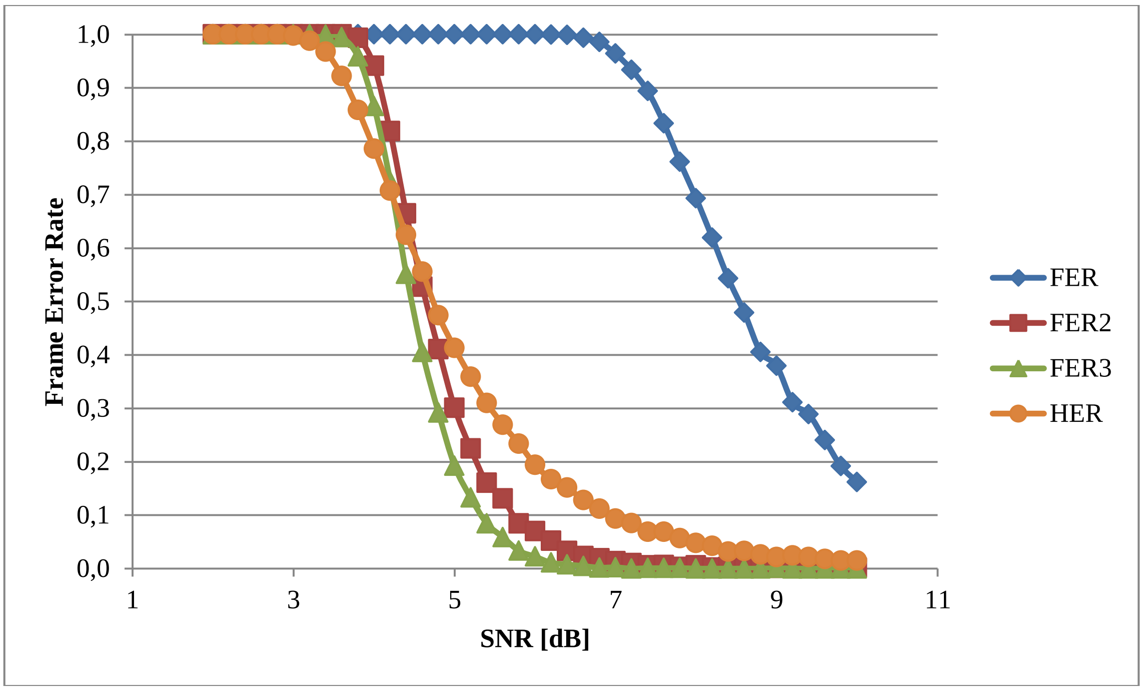


Figure 3-34 Simulation results: TS0, DQPSK, without equalizer

FER vs SNR for DQPSK without using equalizer is plotted in Figure 3-34. By comparing Figure 3-33 and Figure 3-34, it can be observed that the use of the equalizer does not significantly impact the performance. The reason is that the GNU Radio equalizer is a blind LMS DD Equalizer and it is not working properly. This equalizer takes the signal constellation as an input and tries to match the received point to a point in the constellation. This often leads to mistakes.

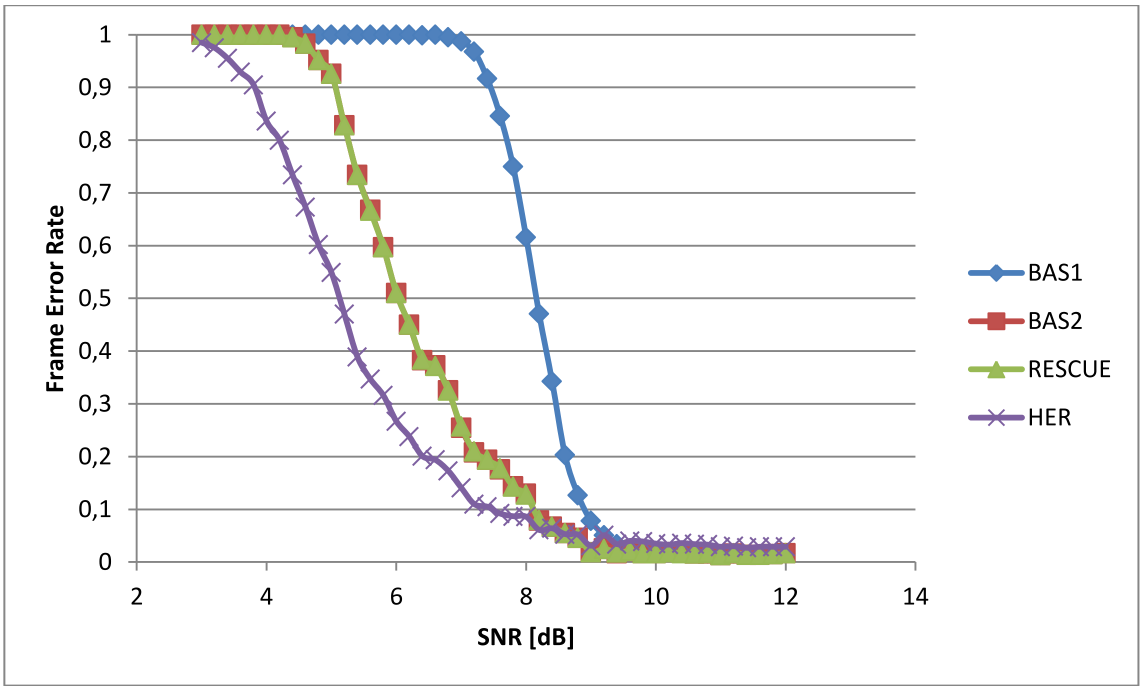


Figure 3-35 Simulation results: TS1, Relay Location A, DQPSK, with LMS DD equalizer

Figure 3-35 shows the FER of baseline 1, baseline 2, RESCUE, and header error rate (HER) in the TS1 for relay location A, respectively. DQPSK and LMS DD equalizer are used at receiver. The BAS1 is the SDF without joint decoder; BAS2 is SDF with joint decoder but without LF. We can see that, RESCUE Lossy Forwarding can achieve better FER performance than the BAS1. This is because the RESCUE converts the relaying system into a distributed turbo code. The advantage of RESCUE is reflected by utilizing the correlation knowledge of the information between source and relay through the joint decoding process. Interestingly, it is found that RESCUE and BAS2 show almost the same performance. The reason is that, in relay location A, the relay is located close to the source, the geometric gain of the intra-link is large and the intra-link error probability is very small. Most of the information sequences are correctly decoded at relay. Therefore, the RESCUE with the erroneous information sequences forwarded by relay and the BAS2 without the erroneous information sequences forwarded by relay achieves almost the same FER performance.

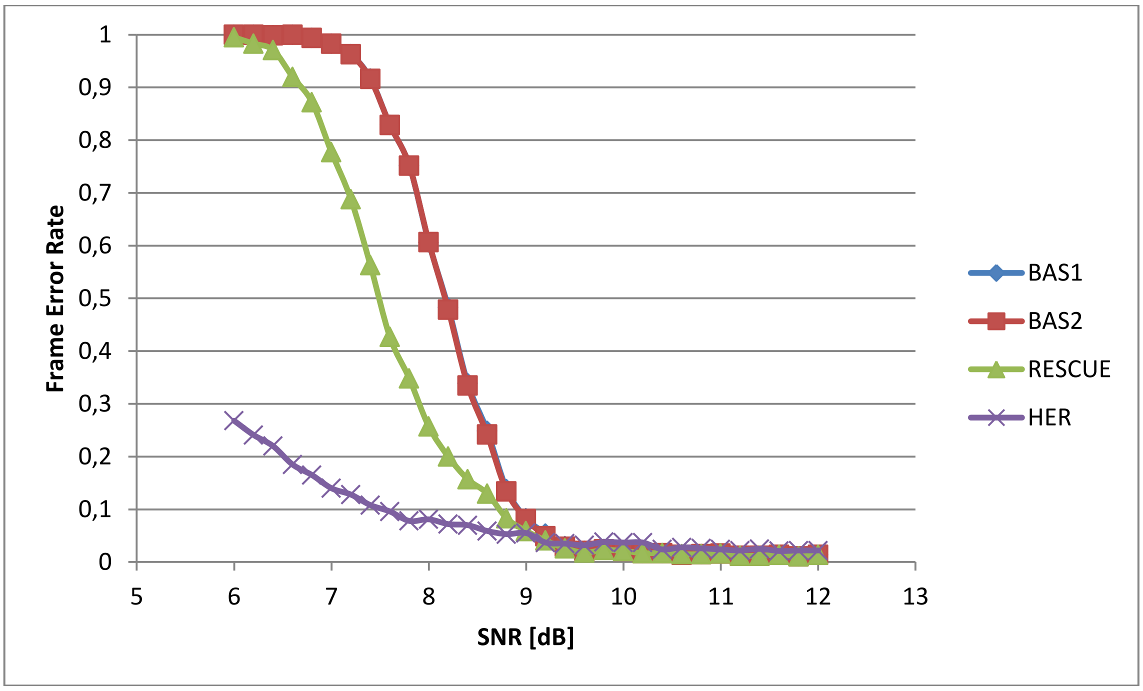


Figure 3-36 Simulation results: TS1, Relay Location B, DQPSK, with LMS DD equalizer

Figure 3-36 compares the FER between the BAS1, BAS2, RESCUE, and HER versus the average SNR for relay location B. Obviously, for both the BAS1 and the BAS2 have almost the same FER performance. However, the FER of RESCUE always achieves better FER performance than both the BAS1 and the BSA2. This observation indicates that when the source is far from the relay, the RESCUE shows its advantage of forwarding the erroneous information sequence and utilizing the correlation knowledge of the information between the source and the relay through the joint decoding process. The BAS2 only jointly decodes the information sequences transmitted from source and the information sequences perfectly decoded and forwarded by relay. Therefore, with the relative high intra-link error probability in relay location B, BAS2 only with the joint decoding scheme could not achieve lower FER performance compared to BAS1, as in relay location A.

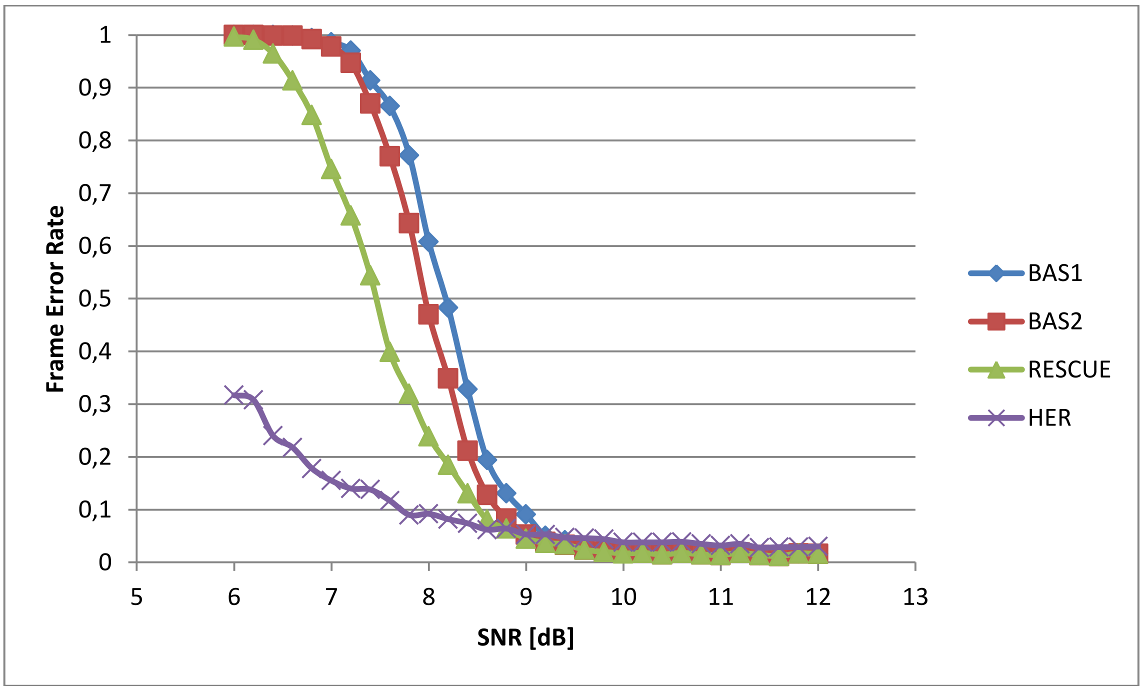


Figure 3-37 Simulation results: TS1, Relay Location C, DQPSK, with LMS DD equalizer

Figure 3-37 presents the FER comparison between the BAS1, BAS2, RESCUE and HER for relay location C. It is found that the FER of BAS2 is smaller than BAS1. This is because that, the relay is located at the position where decoding errors for some information sequences happen in the relay, and the erroneous information sequences are forward to the destination. Destination performs joint decoding in BAS2. Relay location C is the scenario where the RESCUE Lossy Forwarding gain is higher than in the relay location A and smaller than in relay location B.

3.6 Summary of simulation results

In this section, we summarize the simulation results and draw the conclusions that can be made. The first observation that can be made by looking at TS0 results, is that the largest relative gain comes when taking the second copy of the frame (SNR difference between FER and FER2). By taking more copies, the decrement in SNR becomes smaller and smaller. This is reasonable, because decreasing the SNR below a certain value, the decoders of individual branches cannot produce enough information to help each other, and the convergence of the iterative receiver cannot b achieved.

The next observation that we can made from Figure 3-12 and Figure 3-13 is that increasing the length of the payload does not have significant impact on the convergence properties of the iterative receiver. The turbo cliff seems to be slightly steeper with a large payload size compared to the small payload size.

We can also observe in all results, that the requirement of the error-free header restricts the performance of RESCUE. Possible solutions for the problem are

- increase the transmit power of the header,
- have a stronger code (lower rate) for the header,
- decrease header size.

TS1 simulations were performed for three different protocols: BAS1, BAS2, and RESCUE. A comparison between BAS1 and BAS2 reflects the gain induced by joint decoding using RESCUE encoding-decoding strategies. A comparison between BAS2 and RESCUE reflects the gain induced by Lossy Forwarding. We observed that if the relay is placed closer to the destination, it performs equally with SDF, which should be used when successful decoding is available [KGG05]. Lossy forwarding introduced in RESCUE, can be viewed as one form of estimate-and-forward (EF) [cover1979capacity] relaying strategy, which is one suboptimal version of the compress-and-forward (CF) [cover1979capacity] strategy, to be applied when successful decoding is not possible. Hence, by looking at the TS1 results with different relay locations, one can conclude that RESCUE is a hybrid relaying strategy adapting to the environment as follows:

- RESCUE performs as an optimal SDF when successful decoding is possible.
- RESCUE performs as an EF when successful decoding is not possible.

It is clear that this is one of our major practical contributions of finding a way to flexibly adapt between the two relaying strategies according to the observed environment.

It was also observed that in an AWGN channel, the SNR range where RESCUE offers better performance than the baselines is restricted by the very steep turbo cliff in the RESCUE decoder. This means that the SNR range from BER=0.2 to BER=0 is very small. The decay of the BER curve is smaller in fading channels and therefore, the gains of RESCUE are expected to be larger with more realistic channel conditions.

4. Conclusions

This deliverable provided a report on the RESCUE Lossy Forwarding concept implementation and evaluation in a software defined radio platform. First, in section 2, a detailed description of implemented software modules and GNU Radio blocks were presented. This section is divided into three main subsections. In subsection 2.2 a RESCUE source node implementation is presented. Subsection 2.3 covers description of the RESCUE destination node and finally subsection 2.4 presents the relay node.

Section 3 contains simulation results analysis obtained by the use of software defined radio implementation described in section 2. It starts with subsection 3.1 where the simulation software and models are presented. Then, in subsections 3.2 and 3.3 simulation scenarios, parameters and used channel models are described. Subsections 3.4 and 3.5 provide a detailed simulation analysis of the Lossy Forwarding concept in various scenarios and use cases. Finally, subsection 3.6 summarizes the simulation analysis results.

Performance analysis of the RESCUE LF concept developed within WP1 and WP2 in the software defined radio platform allowed to draw important conclusions. First of all, the RESCUE LF technique performs always better than the baseline SDF with or without joint decoder in the destination node.

In case when the relay node is close enough to the source node to establish a lossless radio link (location A) the performance of the RESCUE LF is close to the SDF with joint decoder. Lossless radio link between source and relay implicates no errors at the relay node, so in that case there is no lossy forwarding. However, an important conclusion is that RESCUE coding algorithms do not introduce loss in a non-LF scenarios. This is exactly how it was expected to be according to information theoretical analysis of relay channels.

However, when the distance between source and relay node is too far to establish a lossless radio link (Location B and C) and there are errors in the relay node after decoding performance of the RESCUE LF is significantly better than SDF. In that case RESCUE can be viewed as a form of estimate-and-forward relaying strategy, which is a suboptimal version of the optimal compress-and-forward strategy to be applied when successful decoding is not possible at the relay. This leads to the conclusions that RESCUE LF performs near optimally through the whole range of relay positions and RESCUE coding structure can improve performance of the SDF.

It should be noted that, because of the problem with the implementation of the feedback from the decoder to the demapper in GNU Radio, in all simulation Gray mapping was used. It is expected that optimal mapping with feedback will improve the RESCUE LF performance.

Moreover, software defined radio platform performance analysis have shown that the gain is the biggest when joint decoder in the destination node decodes jointly two copies of the transmitted frame. Sending the third and fourth copy to the destination usually do not bring significant gain.

The most important D2.3 contributions can be summarized as follow:

- implementation of the RESCUE environment, which can be used by the whole GNU Radio community,
- presentation of the RESCUE LF concept simulation results and its performance comparison with SDF joint decoding strategy in many scenarios (QPSK, DQPSK, 16QAM, D16QAM, with and without waveform, AWGN and fading channel)
- feedback from practical verification to theoretical research.

Software defined radio platform performance analysis identified the following problems to solve in our future work:

- Increase header protection, which is required to receive bigger gain when sending more than two copies of the same frame,
- Improve performance of the GNU Radio synchronization and equalization modules in order to enable over the air communication in frequency-selective fading channels

- Introduce training sequence to reception path of the relay and destination nodes in order to improve SNR estimation and enable implementation of non-blind equalizers.

5. References

[anton2014machine]	Anton-Haro, C. and Dohler, M. “Machine-to-machine (M2M) Communications: Architecture, Performance and Applications”, Woodhead Publishing Series in Electronic and Optical Materials, Elsevier Science, 2014.
[cover1979capacity]	T. Cover and A. E. Gamal, “Capacity theorems for the relay channel,” Information Theory, IEEE Transactions on, vol. 25, no. 5, pp. 572–584, 1979.
[ETSI-ITS-G5]	ETSI, Intelligent Transport Systems (ITS); Access Layer; ITS-G5 Channel Models and Performance Analysis Framework (ETSITS 103 257 V0.0.1), Jun. 2014.
[Harris_01]	F. J. Harris, M. Rice, "Multirate Digital Filters for Symbol Timing Synchronization in Software Defined Radios", IEEE Selected Areas in Communications, Vol. 19, No. 12, Dec., 2001.
[HZA+13]	Xin He, Xiaobo Zhou, Khoirul Anwar, and Tad Matsumoto. “Estimation of Observation Error Probability in Wireless Sensor Networks”. In: IEEE Commun. Lett. 17.6 (June 2013), pp. 1073–1076.
[KGG05]	Kramer, Gerhard, Michael Gastpar, and Piyush Gupta. "Cooperative strategies and capacity theorems for relay networks." *Information Theory, IEEE Transactions on* 51.9 (2005): 3037-3063.
[MWHP]	http://www.mathworks.com/
[NIHP]	http://www.ni.com/labview/
[RESCUE D2.1.1]	RESCUE Project, “D2.1.1 - Interim Report of Detailed Coding/Decoding Algorithms and Correlation Estimation Techniques”, Version 1.0, October, 2014
[RESCUE D3.2]	RESCUE Project, “D3.2 – Report on revised WP3 architecture including simulation results”, Version 1.0, July 2015
[RESCUE D4.3]	RESCUE Project, “D4.3 - Report on channel analysis and modelling”, Version 1.0, August 2015
[YG11]	Roua Youssef and Alexandre Graell i Amat. “Distributed Serially Concatenated Codes for Multisource Cooperative Relay Networks”. In: IEEE Trans. onWireless Communications 10.1 (Jan. 2011), pp. 253–263.