MONASH University

Practical Physical Layer Network Coding

Dmitry Kramarev

A thesis submitted for the degree of Doctor of Philosophy at
Monash University in 2016
Department of Electrical and Computer Systems Engineering

Copyright notice

Abstract

PHYSICAL-layer network coding (PNC) is a new technology which has the potential to increase network throughput beyond existing standards based on routing. Despite the fact that PNC has been well investigated from information-theoretic point of view, only a few partial prototypes have been reported in the literature. The implementation of a PNC system is burdened with many challenges such as carrier-phase, symbol and frame asynchrony. In this research, we mainly focus on software-defined radio prototyping of a two-way relay network utilizing PNC relaying. We present the first real-time implementation of a generalized PNC algorithm, namely compute-and-forward relaying. In addition, we propose an improved compute-and-forward relaying scheme which simplifies the use of power-of-two size constellations typically used in practical communication systems. The presented testbed provides a valuable platform for verification of the theoretical research on PNC and evaluation of synchronization requirements. Our experimental results show that when the signal-to-noise ratio is high, PNC relaying outperforms other relaying strategies in terms of the network throughput.

In addition, we propose a new method of multiplierless pulse-shaping filter design which allows essential reduction of the hardware utilization as well as out-of-band power. Therefore, a multiplierless filter designed with the proposed method is especially suitable for FPGA/VLSI implementation.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Dmitry Kramarev
30/03/2016

Publications

Journal Papers

[KSV16] D. Kramarev, A. Sakzad, and E. Viterbo, "Implementation of a two-way relay network with compute-and-forward in GNU Radio," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 4, pp. 484–493, Apr. 2016.

Conference Papers

[KHV14] D. Kramarev, Y. Hong, and E. Viterbo, "Software defined radio implementation of a two-way relay network with digital network coding," in *2014 Australian Communications Theory Workshop (AusCTW)*, vol. 1, Sydney, Australia, Feb. 2014, pp. 120 – 125.

[Kra16] D. Kramarev, "Accurate symbol-level synchronization of universal software radio peripherals for physical-layer network coding applications," Oct. 2016, submitted to IEEE International Conference on Communications 2017.

Acknowledgements

During the course of my PhD studies I have been very fortunate to receive valuable assistance, suggestions and support from many people. I greatly appreciate their generosity in devoting their time and consideration.

First, I am pleased to acknowledge the continuous support, guidance and encouragement of my PhD supervisor, Professor Emanuele Viterbo. This PhD dissertation would not have been possible without his invaluable guidance and persistent help.

I feel extremely fortunate to have worked with members and former members of the Software Defined Telecommunications Laboratory at Monash University. I would especially like to acknowledge the support, collaboration and contribution to our common projects of Dr. Amin Sakzad. I am very thankful for the friendly and helpful environment to Dr. Harshan Jagadeesh, Dr. Shuiyin Liu, Dr. Lakshmi Natarajan and Dr. Sinan Kahraman. I would also like to thank software-defined radio expert Mr. Mikael Eriksson for his constructive discussions and comments.

I would appreciate the help of the ECSE department technical staff Mr. Ray Cooper, Mr. Daryl Gaspero, Mr. Ian Reynolds and others, as well as IT staff Ms. Vanessa Luu and Mr. Godwin Vaz for their kind cooperation in setting up the hardware and software required for my experiments in the lab. I would also express my appreciation to the department staff, Mr. Geoff Binns, Ms. Emily Simic, Ms. Ros Rimington and Ms. Maria Scalzo.

I am thankful for the excellent research supports provided by Australian Federal and Victoria State Governments and the Australian Research Council through the ICT Centre of

Excellence program, National ICT Australia (NICTA).

I would also like to thank Ms. Jane Moodie for her patience when helping me with proofreading and structuring my papers, and Dr. Alex McKnight who assisted by proofreading the final draft of this thesis for grammatical and stylistic errors.

Last but not least, I would like to thank the former Head of the ECSE Department, Prof. Jamie Evans, for his efforts in monitoring my research progress regularly and providing helpful and timely feedback.

Finally, it is my pleasure to acknowledge my family, relatives and friends, whose love, support and encouragement have accompanied me throughout my life.

Contents

List of Tables

List of Figures

List of Acronyms & Symbols

List of Acronyms

5G	5th Generation
ADC	Analog-to-Digital Converter
AF	Amplify-and-Forward
AGC	Automatic Gain Control
ANC	Analog Network Coding
ARQ	Automatic Repeat Request
AWGN	Additive White Gaussian Noise
BER	Bit-Error Rate
BPF	Bandpass Filter
bps	bits per second
BPSK	Binary Phase Shift Keying
CDMA	Code Division Multiple Access
CF	Compute-and-Forward
CIC	Cascaded Integrator-Comb
CLLL	Complex Lenstra-Lenstra-Lovasz
CORDIC	COordinate Rotation DIgital Computer
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSD	Canonical Signed Digit
D2D	Device-to-Device
DAC	Digital-to-Analog Converter
dB	Decibel

dBm	Decibel-milliwatts
DC	Direct Current
DDC	Digital Down-Convertion
DF	Decode-and-Forward
DNC	Digital Network Coding
DQPSK	Differential Quadrature Phase Shift Keying
DSP	Digital Signal Processing
DUC	Digital Up-Convertion
ECC	Error-Correcting Code
FDMA	Frequency Division Multiple Access
FER	Frame Error Rate
FIFO	First In First Out
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
GF	Galois Field
GHz	Gigahertz
GNU	GNU's Not Unix!
GPS	Global Positioning System
GPSDO	GPS Disciplined Oscillator
GPU	Graphics Processing Unit
GRC	GNU Radio Companion
GUI	Graphical User Interface
HB	Half-band
HDL	Hardware Description Language
I/Q	In-Phase/Quadrature
IEEE	Institute of Electrical and Electronics Engineers
IF	Intermediate Frequency
IIR	Infinite Impulse Response
ISE	Integrated Synthesis Environment
ISI	Inter-Symbol Interference
kbps	kilobits per second

LDPC	Low-Density Parity-Check code
LLL	Lenstra-Lenstra-Lovasz
LNA	Low-Noise Amplifier
LO	Local Oscillator
LPF	Low-Pass Filter
LUT	Look-Up Table
MA	Multiple Access
MAC	Multiple Access Control
MBd	Megabaud
MHz	Megahertz
MIMO	Multiple-Input Multiple-Output
MM	Mueller and Muller
MMSE	Minimum Mean-Square Error
MSPS	Mega-Samples Per Second
MUD	Multi-User Detection
NC	Network Coding
NCO	Numerically Controlled Oscillator
NI	National Instruments
NIC	Network Interface Controller
ns	nanosecond
OFDM	Orthogonal Frequency-Division Multiplexing
OOT	Out-of-Tree
PAM	Pulse Amplitude Modulation
PC	Personal Computer
PDR	Packet Delivery Ratio
PGA	Programmable-Gain Amplifier
PHY	PHYsical layer
PNC	Physical Layer Network Coding
PNCF	PNC Over Finite Set
PNCI	PNC Over Infinite Set
PSK	Phase Shift Keying

QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RC	Raised Cosine
RF	Radio Frequency
RRC	Root-Raised Cosine
RS	Reed-Solomon
RT-ARQ	Relay-Terminal ARQ
Rx	Receiver
SDR	Softwared-Defined Radio
SIMD	Single Instruction Multiple Data
SNR	Signal-to-Noise Ratio
SOP	Start of Packet
STRP	Symbol-Timing Recovery Preamble
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
TED	Timing Error Detector
TO-ARQ	Terminal-only ARQ
TS	Traditional Scheduling
TWRC	Two-Way Relay Channel
TWRN	Two-Way Relay Network
Tx	Transmitter
UDP	User Datagram Protocol
UHD	USRP Hardware Driver
UHF	Ultra High Frequency
USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
VHF	Very High Frequency
VLSI	Very Large-scale Integration
VOLK	Vector-Optimized Library of Kernels
XOR	eXclusive or
ZF	Zero-Forcing

List of Symbols

$*$	The convolution operation between $f(t)$ and $g(t)$, $(f*g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$
i	Imaginary unit, $i = \sqrt{-1}$
$\mathbf{A}$	Matrix $\mathbf{A}$
$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$
$\mathbf{A}^*$	Hermitian Transpose of matrix $\mathbf{A}$
$\mathbf{A}^\dagger$	Pseudo inverse of matrix $\mathbf{A}$
$\mathbf{a}$	Vector $\mathbf{a}$
$\|\mathbf{a}\|$	l^2-norm of $\mathbf{a}$
$\|a\|$	The absolute value of a
$\hat{a}$	Estimate of a
$\mathbb{C}$	The set of complex numbers
$\log^+(x)$	$\max(\log_2(x), 0)$
$\Re(\cdot)$	Real part of a complex number
$\Im(\cdot)$	Imaginary part of a complex number
$a == b$	a is equal to b
$a \propto b$	a is proportional to b
$\oplus$	Bitwise XOR
$[\mathbf{v}]_q$	$(v(1) \bmod q, \ldots, v(n) \bmod q)$ for any $\mathbf{v} \in \mathbb{C}^n$.
$\ll$	Logical left shift
E_b/N_0	Energy per bit to noise power spectral density ratio
h_{XY}	Channel coefficient between transmitter X and receiver Y.
$\mathbb{F}_q$	Finite field of order q
$\mathbf{I_n}$	The identity matrix of size n
P	Tx Power
T	Symbol period
T_{out}	Time-out period
$\lfloor x \rfloor$	Floor of x

$z \sim \mathcal{CN}(\mu, \sigma^2)$	z is a circularly distributed complex Gaussian random variable with mean μ and variance σ^2
$\mathbb{Z}[i]$	Gaussian integers
$\mathbb{Z}^+$	Non-negative integers

Chapter 1

Introduction

RELAYING of an important or alarming message was known far before the invention of radio. For example, in the Middle Ages people used drums or bells to notify others about a danger such as enemy intrusion or a coming thunderstorm. When such a message was received, it was repeated by another bell with the same tone and frequency, i.e., relayed in order to propagate the alarm among larger numbers of people. Not surprisingly, relaying in radio communication became a common task soon after the invention of radio. Radio relaying was needed when the source and the destination of a message could not communicate directly due to the large distance, obstacles or power limitations. A relay or a chain of relays simply retransmitted messages without modifying them.

Later, experiments with multiple-input multiple-output (MIMO) wireless systems demonstrated that the quality of the reception could be improved greatly with the use of multiple antennas [1, 2]. This effect was achieved by introducing the diversity, i.e. delivering several copies of the signal via independent paths, thus reducing the negative effects of fading. The subsequent introduction of space-time codes encouraged further development of MIMO systems, improving both diversity and rate without the need for redundant channel use such as repetition of the same signal [3–5]. However, not all wireless devices were able to benefit from the use of transmit diversity because of limited size, power constraints or hardware costs. To overcome this, the concept of *cooperative communication* [6] was introduced. In this scenario, several single-antenna devices in a network share their antennas in order to create a virtual MIMO system. The introduction of cooperative communication allowed single-antenna devices to exploit transmit diversity, thus

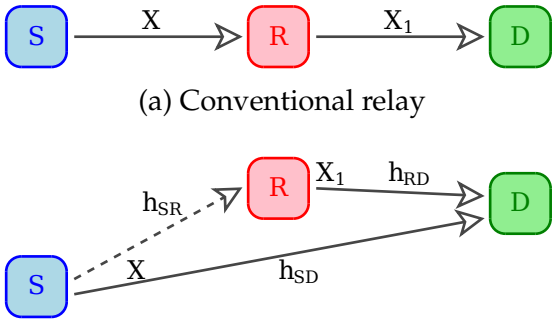


Figure 1.1: Difference between (a) conventional relaying and (b) cooperative relaying. The conventional relay R simply retransmits message $\mathbf{x}_1$ received from source S transmitted $\mathbf{x}$ toward destination D. In the cooperative relaying scheme source S first transmits $\mathbf{x}$ to relay R; then both R and S transmit $\mathbf{x}$ simultaneously in order to take advantage of transmit diversity.

increasing the quality of communication. Accordingly, cooperative communication introduced a new role for relays. In this scenario, the task of relaying changed from merely repetition to cooperative relaying. Figure 1.1 illustrates the difference between conventional and cooperative relaying in wireless communication systems.

In large wireless networks, such as mobile networks, the traffic is often bidirectional, when a node can serve as a source and sink at the same time, or even multidirectional. A *two-way relay network* (TWRN), alternatively referred to in the literature as *two-way relay channel* (TWRC), represents a canonical example of a network topology with bidirectional relaying. TWRN represents a wireless network consisting of three nodes, namely terminals A and B and relay R. Since there is no direct communication link between A and B, the terminals exchange information via the relay R. We assume that the nodes operate in half-duplex mode, i.e. a node can either receive or transmit at the same time, but not both. A traditional scheduling (TS) scheme shown in Figure 1.2 exchanges information in

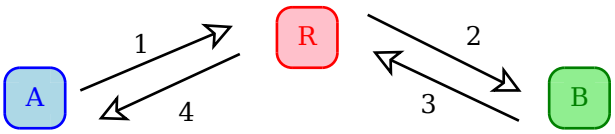


Figure 1.2: Two-way relay network with traditional scheduling scheme. The information exchange is performed within four time slots.

Figure 1.3: Two-way relay network with digital network coding. The information exchange is performed within three time slots.

TWRN in four time slots. First, terminal A transmits a packet $\mathbf{m}_A$ to relay R; second, relay R forwards $\mathbf{m}_A$ to terminal B; at the third and fourth time slots, respectively, B transmits a packet $\mathbf{m}_B$ to relay R, and R forwards $\mathbf{m}_B$ to A. This example illustrates that the relay transmits twice as often as the terminals, thus the relay represents a bottleneck in the network. This mean that with growing demands for higher data rates, this approach is unable to maximize the network throughput [7]. Therefore, introducing a more efficient relaying strategy can lead to throughput increase of the entire network. For example, the relay may need to compute functions of packets it receives, and forward the functions, rather than the original packets.

With *digital network coding* (DNC), presented in Figure 1.3, it is possible to reduce the number of time slots in TWRN information exchange from four to three as follows: in the first and second time slots each terminal transmits its packet $\mathbf{m}_A$ and $\mathbf{m}_B$, respectively, to the relay; using conventional demodulation techniques, the relay independently recovers bits of both the received packets, performs bitwise XOR operation and composes packet $\mathbf{m}_R$ such that

$$\mathbf{m}_R = \mathbf{m}_A \oplus \mathbf{m}_B, \tag{1.1}$$

where $\oplus$ denotes bitwise XOR. In the third time slot, relay R broadcasts packet $\mathbf{m}_R$ to the both terminals. Subsequently, each terminal is able to recover the packet addressed to it by performing another XOR operation on $\mathbf{m}_R$ and its own transmitted packet:

$$\mathbf{m}_R \oplus \mathbf{m}_A = (\mathbf{m}_A \oplus \mathbf{m}_B) \oplus \mathbf{m}_A = \mathbf{m}_B \tag{1.2}$$

As a result, by performing data exchange in three time slots rather than in four, DNC increases TWRN throughput by 33%.

DNC was first proposed for wired networks in [8]. Later, [9,10] developed the impor-

tant class of linear network codes, where $\mathbf{m}_R$ is computed as a linear combination of $\mathbf{m}_A$ and $\mathbf{m}_B$ over a finite field GF (q)

$$\mathbf{m}_R = \left[a_A \mathbf{m}_A + a_B \mathbf{m}_B\right]_q \tag{1.3}$$

where a_A and a_B are coefficients over the finite field. From the perspective of these works, the DNC scheme with bitwise XOR operation, described in (1.1) and (1.2), can be considered as a special case over the finite field GF (2). Subsequently, DNC was applied to wireless networks in [11] and its performance was investigated in [12, 13]. In particular, [13] proposes a DNC scheme where the relay forwards the soft decisions, i.e. the soft-DNC scheme. The simulation results provided demonstrate that the soft-DNC scheme is especially useful when Tx SNR is high.

Typically, a relay utilizes one of the MAC methods, such as TDMA or FDMA, or multiuser detection in order to avoid interference from several users. However, these conventional methods do not allow further increase of the throughput. One way of boosting the throughput derives from the observation that a relay is often not interested in the content of a packet which it receives and forwards.

Physical layer network coding (PNC) represents a further development of the network coding concept, where network coding is performed in the physical layer. By exploiting interference rather than avoiding it, the PNC in theory may double the network throughput. Unlike DNC, where the network coding operations are performed explicitly by the relay, the PNC concept proposed independently in [14], [15] and [16] in 2006 shifts the burden of network coding from the relay to the wireless channel. Specifically, PNC allows several nodes to transmit synchronously. As a result, superposition of electromagnetic waves occurs in the wireless medium, and this phenomenon is exploited as a form of network coding, performed by nature. In this way, the interference effect, which is usually seen as destructive, becomes constructive. Note that early theoretical works on PNC assume perfect synchronization between several terminals. Subsequent studies, however, have demonstrated that the synchronization requirements can be significantly alleviated. The needs for synchronization are discussed in detail in Section 2.4.

With respect to TWRN, PNC allows for further reduction of the number of time slots

Figure 1.4: Two-way relay network with PNC. The information exchange is performed within two time slots. The first and second time slots are referred to as the multiple-access phase and the broadcast phase respectively.

to two by combining transmissions from both the terminals into one time slot, as illustrated in Figure 1.4. Let signals $x_A(t)$ and $x_B(t)$ respectively represent packets $\mathbf{m}_A$ and $\mathbf{m}_B$ after modulation. If the signals are transmitted synchronously within the first time slot, in this case also referred to as the *multiple-access* (MA) phase, the relay receives the superimposed signal:

$$y(t) = h_A x_A(t) + h_B x_B(t) + z(t), \tag{1.4}$$

where h_A and h_B represent channel coefficients, and $z(t)$ is channel noise.

Subsequently, based on the received signal $y(t)$, the relay attempts to estimate vector $\mathbf{m}_R$

$$\widehat{\mathbf{m}}_R = F\left(\mathbf{m}_A, \mathbf{m}_B\right) = f\left(y\left(t\right)\right), \tag{1.5}$$

where $F\left(\cdot,\cdot\right)$ is referred to as *PNC mapping* function, and $f\left(y\left(t\right)\right)$ determines the mapping of $y\left(t\right)$ to $\widehat{\mathbf{m}}_R$.

During the second time slot, the *broadcast phase*, the relay broadcasts $\widehat{\mathbf{m}}_R$. Upon reception, the first terminal recovers the desired packet $\widehat{\mathbf{m}}_B$ based on $\widehat{\mathbf{m}}_R$ and its own transmitted packet $\mathbf{m}_A$, and the second terminal recovers $\widehat{\mathbf{m}}_A$ in the same way. The following example from [17] explains the principle behind the PNC.

Let $h_A = h_B = 1$, and both the terminals transmit just one bit modulated with BPSK modulation. We assume that the channel is noiseless, and the terminals transmit with the same power and at the same time. Then

$$y(t) = x_A(t) + x_B(t). \tag{1.6}$$

All possible combinations are shown in Table 1.1. From the table it follows that in this

Table 1.1: Example explaining principles of PNC [17].

Bit of m_A	Bit of m_B	Symbol of $x_A(t)$	Symbol of $x_B(t)$	Received symbol $y(t)$	$m_A \oplus m_B$	Bit of $\widehat{m}_R$
0	0	-1	-1	-2	0	0
0	1	-1	1	0	1	1
1	0	1	-1	0	1	1
1	1	1	1	2	0	0

example bitwise XOR can be employed as the PNC mapping function:

$$F\left(m_A, m_B\right) = m_A \oplus m_B, \tag{1.7}$$

and

$$f\left(y\left(t\right)\right) = \begin{cases} 1, & |y\left(t\right)| = 0, \\ 0, & |y\left(t\right)| = 2. \end{cases} \tag{1.8}$$

Subsequently, the relay broadcasts $\widehat{m}_R$ and the terminals A and B receive it and recover respectively

$$\widehat{m}_B = \widehat{m}_R \oplus m_A = m_B, \tag{1.9}$$

and

$$\widehat{m}_A = \widehat{m}_R \oplus m_B = m_A. \tag{1.10}$$

From this example, the term "physical layer network coding" is straightforward: network coding in this scenario naturally happens in the physical layer. This example also demonstrates that in PNC, there is no need for the relay to recover the bits of $\mathbf{m}_A$ and $\mathbf{m}_B$ from $y(t)$; only the bits of $\mathbf{m}_R$ are required. Accordingly, PNC increases TWRN throughput by 100% compared to TS, and 50% compared to DNC. This simple example is, however, based on unrealistic assumptions. In practice, the throughput can drop depending on channel conditions and the Tx power. Subsequent studies of PNC propose and develop algorithms applicable to realistic situations, higher order modulations such as QPSK and M-QAM, and analyze their performance.

Despite the potential of PNC to boost the network throughput and promising theo-

retical results, it is still in the early stage of development from the perspective of its use in actual wireless communication systems. In addition, its implementation is hampered by numerous implementation problems. Therefore, the development of PNC into a technology suitable for utilization in the next generation telecommunications, its prototyping and experimental evaluation represent the research foci of this thesis.

1.1 Research motivation

In spite of the fact that many theoretical studies and simulation results have been published, little research has been reported on the deployment of PNC in wireless communications. In particular, limited results are available on the prototyping of PNC algorithms, especially on real-time hardware. Currently, only a few simplified PNC testbeds are implemented. They support low data rates due to either high computational complexity or insufficient synchronization, or both. Therefore, they do not fully demonstrate all the benefits of PNC. The reader is referred to Section 2.6 for a detailed overview of the implementation of PNC testbeds available in the literature.

At the same time, the number of reported DNC implementations is significantly higher. DNC has been applied in various areas such as routers for mesh networks [18], video streaming on smartphones [19–21], or implemented on software-defined radio (SDR) [22]. In all cases, the utilization of network-coding led to a significant increase in the network throughput. For example, [20] demonstrated that cooperation and the exchange of network-coded messages among several smartphones located within proximity of each other increased the average download rate for each smartphone, when downloading the same video content from the Internet, by up to three times. In [18], Katti *et al.* present a new architecture for wireless mesh networks (COPE), which provides improvement of throughput of the network by up to four times, achieved by forwarding DNC-mixed packets. The testbed consists of 20 wireless nodes located on two floors of a building.

Since in theory PNC outperforms DNC in terms of throughput, we believe that the practical benefits of PNC will be significant. In our opinion, PNC has large potential for applications in various areas. From this discussion and taking into account the gap

between theory and practice, we consider the design of a stable real-time PNC testbed and subsequent experimental, rather than theoretical analysis, to be the first step toward demonstrating the full potential of PNC.

Such a testbed, if implemented, will provide inestimable assistance in verifying recent theoretical research on PNC. As PNC is a broad research area which attracts researchers with different backgrounds and has a large number of possible applications, we understand the need for the testbed to be universal and quickly modifiable, and to support different data rates and allow experiments with different frequencies, modulations, channel coding schemes and other techniques. Therefore, SDR appears to be a good basis for implementation of such a testbed. The reader is referred to Section 2.7 and references therein for more information on the advantages of prototyping communication systems in SDR.

With our main work on prototyping of PNC schemes on existing SDR platforms, we have found that the implementation of a few essential components of a communication system on FPGA, rather than on a general purpose CPU, dramatically improves the performance of the entire system. However, due to the limited number of multipliers available on many commercial FPGAs, the design of those components is preferably multiplierless. An alternative solution would require the use of more expensive FPGAs, which are equipped with multipliers but considerably increase the costs of both design and development. Therefore, multiplierless design remains of interest. In particular, relocation of the root-raised cosine filter to the FPGA used in both Tx and Rx chains for pulse-shaping can greatly increase the throughput of the SDR platform and decreases the randomness of processing delay. Therefore, this motivated us to search for an efficient multiplierless implementation of the root-raised cosine filter on FPGA. In particular, we are looking at infinite impulse response (IIR) filter design, because IIR filters often achieve desired specifications with lower computational complexity, thus reducing the hardware utilization.

1.2 Challenges of PNC implementation

The implementation of PNC systems faces the following major technical challenges which hinder prototyping and subsequent performance evaluation.

- **Information-theoretic focus of the previous research on PNC:** In general, studies on PNC investigate its performance from an information-theoretic point of view. However, many existing PNC algorithms demonstrate relatively poor performance when used directly in scenarios typical for wireless communication systems. For instance, analog PNC schemes are convenient for analysis, and hence are well-studied, but may be not suitable for use in systems based on digital communications.

- **Increased effective noise of PNC receivers:** In addition to channel noise, PNC mapping introduces extra artificial noise. For example, in the compute-and-forward (CF) strategy, effective noise is added due to the need to approximate non-integer complex channel coefficients with the integer coefficients.

- **Symbol and frame synchronization:** Original PNC algorithms require tight symbol and frame synchronization. On the other hand, recent theoretical works on asynchronous PNC are based on assumptions which are not realizable on existing hardware, for example, due to excessive oversampling requirements, or assumptions which lead to entire system inefficiency.

- **Applicability of established signal processing methods:** Widely used signal processing techniques, aiming to improve the quality of the received signal, such as channel estimation and symbol-timing recovery, are well-developed for traditional communications which avoid interference. However, their applicability to PNC systems is unclear.

- **Uncertain delay of SDR platforms:** SDR playing an increasing role in prototyping of next-generation communication systems. However, despite their well-known benefits, SDR systems typically introduce additional random delay, which is hardly predictable, due to the fact that the computations are performed on general purpose

CPUs rather than on dedicated hardware. Therefore, the implementation of PNC on SDR represents additional challenges due to its sensitivity to the asynchrony.

- **Conventional MAC protocols are inefficient when applied directly in PNC networks:** The relay in PNC systems recovers the functions of packets, rather than the packets individually. Therefore, the PNC relay has limited capability to detect damaged packets, which may result in forwarding corrupted packets to the terminals. This factor can contribute to the reduction of effective network throughput. Accordingly, MAC protocols utilized in relaying need to be more sophisticated in order to accommodate the increased role of terminals in such terminal-only data integrity verification.

The impact of all these problems may outweigh the potential positive effect of PNC. The challenges are discussed in more detail in Chapter 2.

1.3 Objectives of the research

According to our research motivations and the challenges, we define the objectives of our research as follows:

- **Objective 1:** Examination of different PNC algorithms for their efficiency in scenarios typical for existing wireless communication standards and the simplicity of their implementation on SDR.
- **Objective 2:** Development of a practical CF scheme based on realistic assumptions and suitable for use in actual communication systems.
- **Objective 3:** Development of MAC protocols capable of supporting CF relaying and robust against GNU Radio inadequacies and USRP hardware imperfections.
- **Objective 4:** Implementation of the first real-time prototype of TWRN with synchronous CF relaying in GNU Radio. This also includes the implementation of a practical symbol and frame synchronization scheme and solving other implementation challenges, such as the symbol-timing recovery and channel estimation.

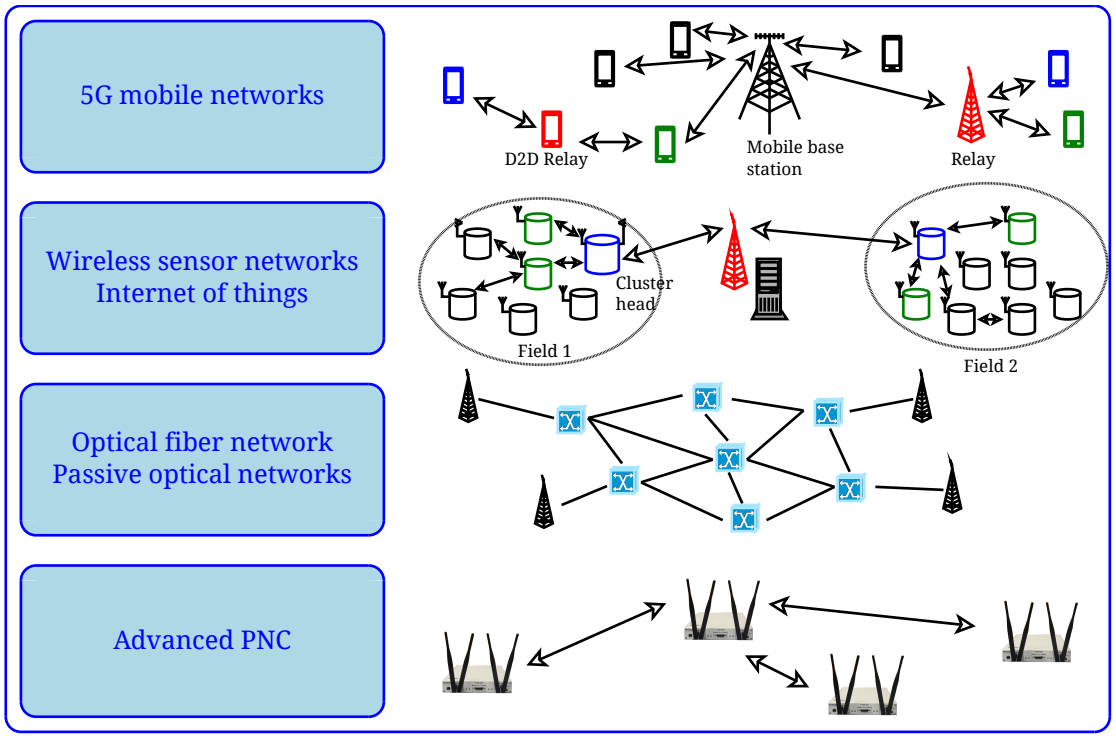


Figure 1.5: Applications which may benefit from PNC.

- **Objective 5:** Experimental verification of the benefits of CF. Our aim is the experimental performance evaluation of CF based on a real-time testbed and comparison of the CF relaying scheme with other non-PNC relaying strategies.

- **Objective 6:** Development of a new method of multiplierless pulse-shaping IIR filter design suitable for implementation on FPGA/VLSI with reduced hardware cost and on wider class of FPGAs without dedicated hardware multipliers.

1.4 Potential applications of PNC

PNC may be useful in many areas where collaborative wireless communications and networking are utilized, and which demand increased network throughput. These applications include, but are not limited to 5G mobile communications, ad-hoc and wireless sensor networks, and Internet access, including Wi-Fi and the Internet of Things. Figure 1.5 shows some applications which will potentially benefit from PNC.

Preliminary discussions on possible standards for 5G mobile networks [23, 24] suggest that the device-centric architecture of a cellular network may be preferred over the conventional base-station-centric architecture. Such architecture allows direct device-to-device (D2D) communication and possibly cooperative communication. In this scenario, the relaying represents a natural expansion of the D2D scheme and may benefit in both increasing throughput and saving energy. The quality of such relaying can be further improved with PNC utilized in the device serving as a relay.

In wireless sensor networks, long-distance direct communication is undesirable because the power budget of sensor nodes is extremely limited, but the energy spent for communication is a superlinear function of the transmission distance. In this scenario, communicating via relay nodes within a short distance greatly expands the network lifetime [25]. In this case, PNC relaying can effectively reduce the amount of data forwarded by the intermediate nodes, thus further reducing energy consumption and increasing the lifetime.

Furthermore, the use of PNC can be extended from wireless networks to optical fiber networks, in particular passive optical networks [17, 26, 27]. For example, [17] provides an example which demonstrates that the throughput of a passive optical network can be potentially doubled when optical PNC is employed.

Finally, the ability for quickly prototype new PNC algorithms can be beneficial for researchers working on PNC. Currently, since many PNC algorithms are verified only in simulations, their actual performance and applicability are questionable. Therefore, the development of a basic TWRN testbed supporting PNC relaying on an SDR platform will simplify design and experimental evaluation of new PNC algorithms.

1.5 Thesis organization

This thesis consists of seven chapters. The thesis organization and relationship between chapters is illustrated in Figure 1.6. The main contributions of this research are reported in Chapters 3 to 7 as follows:

Chapter 2. This chapter introduces different concepts of PNC in more detail, including

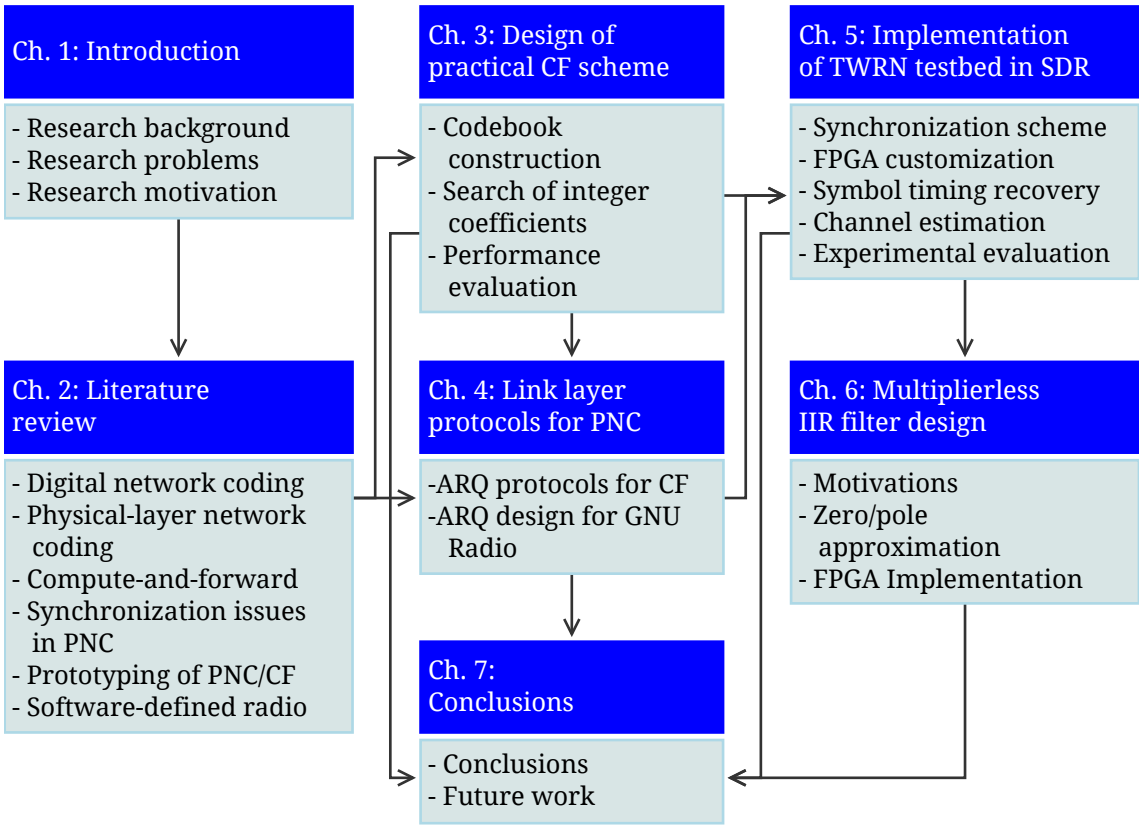


Figure 1.6: Thesis structure and relationship between chapters.

the CF approach. We then provide an extensive literature review of recent results on PNC and CF, and compare these schemes with other relaying strategies, traditional and those based on non-PNC network coding. We also review recent prototyping efforts of various network coding schemes available in the literature. In the second part, we briefly introduce the concept of SDR and provide some examples of recent experimental communication systems, the prototyping and experimental evaluation of which was conducted entirely on SDR. We finally introduce GNU Radio, a particular SDR platform utilized for prototyping in this project, and the hardware used by GNU Radio, namely Ettus Universal Software Radio Peripheral (USRP) [28].

Chapter 3. In this chapter we focus on the development of a CF relaying scheme suitable for practical implementation. We propose the CF scheme, which allows the use of

conventional constellations and channel codes such as QPSK and Reed-Solomon (RS) code. At the same time, the scheme provides invertibility of the linear coefficient matrix, thus overcoming the drawbacks of the original CF scheme. Subsequently, we evaluate the performance of the proposed scheme, and compare it with the performance of the original CF scheme and other non-PNC relaying strategies.

Chapter 4. This chapter discusses design in the layers above the physical layer for the CF scheme implementation, in particular the data-link layer. We propose an ARQ protocol which supports the use of PNC and CF relaying in TWRN. At the same time, the ARQ protocol is designed to handle certain features of GNU Radio typical for many SDR platforms, such as random delay. We subsequently discuss implementation features of the proposed protocol in GNU Radio, and compare the implementation with that of protocols utilized for other non-PNC relaying strategies, including the DNC scheme. Finally, simulation results, which compare the throughput of the TWRN with different relaying schemes when they are combined with relevant ARQ protocols, are provided and discussed.

Chapter 5. In this chapter we discuss PNC implementation challenges, and the methods we used to solve them in GNU Radio and for USRP. While some can be resolved in software (C++, Python) entirely, others require a modification of the field-programmable gate array (FPGA) image for the USRPs utilized in the testbed. We then provide a description of the implemented testbed and demonstrate its performance with a number of experiments in the indoor environment. We finally analyze the experimental results and compare the performance of the CF relaying scheme with other relaying strategies, and elaborate on the applicability of CF/non-PNC strategies in different SNR regimes.

Chapter 6. In this chapter we focus on the multiplierless design of IIR pulse-shaping filters and their successive FPGA implementation. We propose a new method of designing IIR filters based on zero/pole approximation. Subsequently, we consider the implementation of a pulse-shaping filter with the proposed method and analyze the accuracy of our approach. Our experimental results demonstrate that

IIR pulse-shaping filters introduce a marginal increase of BER while having significantly lower complexity thus saving hardware resources. In addition, the proposed method allows the implementation of pulse-shaping filters on a cheaper class of hardware not equipped with expensive built-in hardware multipliers.

Chapter 7. This chapter provides conclusions and discusses potential directions for future work.

Chapter 2
Literature review

Since its introduction in 2006, PNC has attracted significant research attention, resulting in a number of promising theoretical results. In this chapter we provide a literature review on the major theoretical developments on PNC. We specially draw our attention to performance analysis of PNC scheme in presence of symbol and frame asynchrony. We also survey the recent efforts on prototyping of PNC relaying. In addition, we provide a brief introduction to the concept of software-defined radio (SDR) and GNU Radio, the SDR platform employed in this project.

2.1 Introduction

THE purpose of this chapter is two-fold. In the first part of this chapter we provide a comprehensive review of the most notable studies on PNC. The second part is dedicated to a survey of the SDR concept, in particular GNU Radio. In this part we also overview the recent contribution of SDR in prototyping new wireless communication technologies.

This chapter is organized as follows. In Sections 2.2 and 2.3 we review the major results on PNC and its generalization, the compute-and-forward (CF) scheme. Section 2.4 summarizes the existing solutions to the problem of symbol and frame asynchrony, which naturally occurs when several terminals are required to transmit simultaneously. Section 2.5 describes the channel estimation problem in PNC. Section 2.6 complements our theoretical review with an overview of implementation efforts on PNC relaying schemes. In Section 2.7 we review the concept of SDR and the advantages of prototyping communication systems on SDR platforms. In particular, we describe GNU

Radio, the SDR platform utilized in this project. We overview our hardware, namely the USRP and its capabilities in Section 2.8. Finally, Section 2.9 concludes this chapter.

2.2 Early studies on PNC

The example provided in Table 1.1 illustrates just one possible PNC mapping scheme, where $\mathbf{m}_R = \mathbf{m}_A \oplus \mathbf{m}_B$. In general, however, the optimal PNC mapping scheme may be different. Furthermore, the symbols of $\mathbf{m}_R$ may belong to a different constellation compared to the symbols of $\mathbf{m}_A$ and $\mathbf{m}_B$. For example, [14] considers a case where all symbols $\mathbf{m}_A$, $\mathbf{m}_B$ and $\mathbf{m}_R$ are from QPSK constellation. In contrast, work [29] shows that in the case when a certain phase mismatch is presented, and the QPSK modulation is used by the terminal, it is not the best solution to employ QPSK for representation of $\mathbf{m}_R$. The paper suggests that a PNC map with at least five constellation points (5-QAM) is needed to ensure the successful decoding at the terminals. Furthermore, in some studies $\mathbf{m}_R$ takes values from an infinite set [30,31]. In case, $\mathbf{m}_R$ consists of symbols from a finite set, such PNC mapping is referred to as PNC over finite set (PNCF). Alternatively, if $\mathbf{m}_R$ takes values from an infinite set, such PNC mapping is referred to as PNC over infinite set (PNCI). According to [30], PNCF schemes usually outperform PNCI when the MA phase is good and the broadcast phase is noisy. However, when the MA phase is noisy, but the broadcast phase is not, the PNCI demonstrates better performance compared to the PNCF. The early works also assumed that the terminals were fully synchronized, including the frequency, symbol and frame synchronization.

In addition to the PNC schemes where the relay recovers $\mathbf{m}_R$ symbol-by-symbol, the relay may also attempt to recover the analog signal

$$x_R(t) = \widehat{F}_A\left(x_A(t), x_B(t)\right) = f_A\left(y\left(t\right)\right), \tag{2.1}$$

where $F_A\left(\cdot,\cdot\right)$ represents the analog PNC mapping function. PNC schemes with analog PNC mapping are referred to as *analog PNC* (ANC). A simple example of ANC is the amplify-and-forward (AF) scheme considered in [32]. In the AF scheme, the relay simply

amplifies the received signal in (1.4), as follows:

$$x_R(t) = g \cdot y(t) = g \cdot (h_A x_A(t) + h_B x_2(t) + w(t)), \tag{2.2}$$

where g is amplification gain. The AF strategy is simple from the relay point of view, does not require tight synchronization, and therefore is easier to implement. On the other hand, it is obvious from (2.2), that the relay also amplifies the unwanted noise. As a result, this strategy imposes more burden on terminals which have to compensate channel effects before subtracting their own part from the received signal. In addition, in [7] the authors notice that the tradeoff is an architectural price which appears when an analog part is embedded into a digital communication system. This is especially important if the relay is to be implemented in software on an SDR platform. In addition, the integration of the AF relaying with existing channel coding schemes which suit digital communication is a difficult task [17].

The compress-and-forward strategy [33,34] is a practical modification of the AF scheme, where the received signal $y(t)$ is quantized into several discrete levels before being relayed, i.e.

$$x_R(t) = g\Delta\lfloor\frac{y(t)}{\Delta}\rfloor, \tag{2.3}$$

where Δ is the quantization step size. The compress-and-forward scheme performs as poorly as the AF scheme, because the relay does not remove the noise from the quantized signal, but simply amplifies and forwards it.

Therefore, PNC relaying schemes removing noise from the relayed signal are more feasible in modern communication systems, compared to the AF or compress-and-forward schemes. One such scheme, the denoise-and-forward (DNF) is proposed in [35]. The authors show that the XOR mapping is not optimal for all channel conditions and subsequently propose an alternative PNC mapping scheme. The proposed scheme does not require phase adjustment between the terminals, i.e. it does not impose constraints on the phase difference between two terminals. However, since the DNF scheme introduces only per-symbol denoising, without per-codeword denoising, channel coding cannot be incorporated in the scheme and should be implemented separately. In addition, the con-

struction and performance of denoising PNC mappings for higher order constellations such as M-QAM are studied in [36, 37].

Despite the fact, that PNC takes two time slots, while DNC takes three time slots for message exchange in TWRN, in certain channel conditions and SNR regimes, DNC relaying can outperform PNC schemes. An extensive comparison of DNC and PNC in TWRN in terms of BER and network throughput with regard to different fading conditions and SNR is provided in [38]. However, due to the simplicity of the derivations, only the AF PNC relaying is considered.

Once the optimal constellations are selected for a PNC relaying scheme, it should be integrated with channel coding in order to enhance reliability of the relaying. A few such ECC schemes in PNC are proposed in [39, 40], and the surveys are provided in [7, 17].

Finally, studies on PNC were extended to MIMO PNC and space-time coding PNC in [41–44].

2.2.1 Alternatives to PNC

We should also discuss other relaying strategies, alternative to PNC, which allow message exchange in TWRN in two time slots. In general, these strategies are referred to as *decode-and-forward* (DF) or *joint DF* in different studies. In DF, the relay decodes packets $\mathbf{m}_A$ and $\mathbf{m}_B$ individually, and then produces packet $\mathbf{m}_R = \mathbf{m}_A \oplus \mathbf{m}_B$. There are a number of different DF schemes.

First, if the relay is MIMO-capable, and the terminals are synchronized, they can transmit their packets simultaneously (MA phase). In this way, the relay receives the following signals:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \mathbf{H} \begin{bmatrix} \mathbf{x}_A \\ \mathbf{x}_B \end{bmatrix} + \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} \tag{2.4}$$

where

$$\mathbf{H} = \begin{bmatrix} h_{A1} & h_{B1} \\ h_{A2} & h_{B2} \end{bmatrix} \tag{2.5}$$

and $h_{A\{1,2\}}, h_{B\{1,2\}} \in \mathbb{C}$ are the channel coefficients estimated at the first and second antenna of the relay, and $\mathbf{z}_{\{1,2\}} \in \mathbb{C}$ is the additive noise. Using MIMO demodulation

techniques such as the zero-forcing (ZF) receiver [45], the relay recovers $\mathbf{x}_A$ and $\mathbf{x}_B$

$$\begin{bmatrix} \hat{\mathbf{x}}_A \\ \hat{\mathbf{x}}_B \end{bmatrix} = \mathbf{H}^{-1} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \tag{2.6}$$

and subsequently demodulates $\mathbf{m}_A$ and $\mathbf{m}_B$. In the second time slot the relay broadcasts $\mathbf{m}_R$. Alternatively to the ZF receiver, other MIMO demodulation techniques such as the minimum mean-square error (MMSE) receiver or the integer-forcing linear receiver proposed in [46] can be employed. The use of the integer-forcing receiver in TWRN, namely the integer-forcing-and-forward scheme, is presented in [47].

In addition to the use of MIMO, other techniques such as multi-user detection (MUD) or CDMA can be utilized for the individual recovery of both packets. In all cases, DF schemes require greater hardware resources, such as multiple antenna RF front-ends and computational resources for multiple signal recovery. Furthermore, DF schemes are often redundant compared to PNC schemes, because the relay recovers each individual packet completely, although only the knowledge of $\mathbf{m}_A \oplus \mathbf{m}_B$ is sufficient to recover the desired messages at the terminals. However, this redundancy may be beneficial in the MAC layer. For example, algorithms such as the CRC can be applied in the DF scheme where the packets are represented in bytes. The CRC algorithm enables the post-ECC verification of data integrity in the DF relay, thus allowing the relay to detect corrupted packets.

2.3 Compute-and-Forward relaying scheme

Compute-and-forward (CF) relaying [48] represents a generalization of early PNC studies. Let N single-antenna terminals transmit simultaneously their length-k packets $\mathbf{m}_i \in \mathbb{F}_q^k$, where q is prime and $i = 1, \ldots, N$, using the same encoder $\mathcal{E} : \mathbb{F}_q^k \to \mathbb{C}^n$, i.e. $\mathbf{x}_i = \mathcal{E}(\mathbf{m}_i)$. Each of M single-antenna relays receives the following signal

$$\mathbf{y}_m = \sum_{i=1}^{N} h_{im}\mathbf{x}_i + \mathbf{z}_m, \tag{2.7}$$

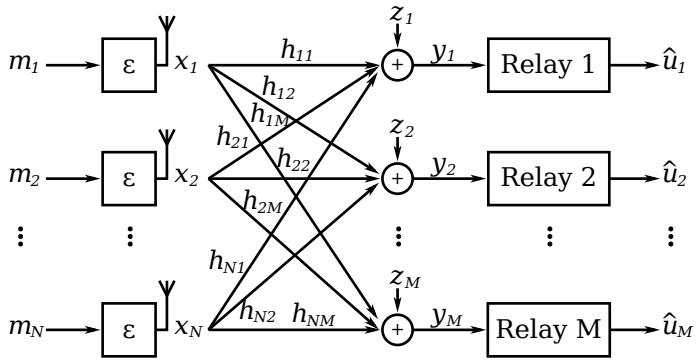


Figure 2.1: System diagram of CF relaying in Gaussian network. N nodes simultaneously transmit

where $m = 1, \ldots, M$, $h_{im} \in \mathbb{C}$ are the channel coefficients, and $\mathbf{z} \sim \mathcal{CN}(0,1)$ is the complex Gaussian noise. Rather than recovering each packet individually, each relay attempts to recover an integer linear combination of the transmitted signals,

$$\mathbf{u}_m = \sum_{i=1}^{N} a_{im}\mathbf{x}_i, \tag{2.8}$$

where $a_{im} \in \mathbb{Z}[i]$. Given $\mathbf{x}_i$ are codewords of $\mathcal{E}$, their linear combination $\mathbf{u}_m$ is again a codeword, hence it can be efficiently decoded. The choice of coefficients a_{im} depends on the channel coefficients h_{im} and the transmit power. Each relay chooses the coefficients a_{im} in order to maximize the computation rate [48] given by

$$R_m^{comp}(\alpha_m, \mathrm{P}, \mathbf{a}_m) = \log^+\left(\frac{\mathrm{P}}{|\alpha_m|^2 + \mathrm{P}\|\alpha_m\mathbf{h}_m - \mathbf{a}_m\|^2}\right), \tag{2.9}$$

where P is the power of the transmitted signal, $\mathbf{h}_m = [h_1, \ldots, h_N]^T$ and $\mathbf{a}_m = [a_1, \ldots, a_N]^T$. Coefficients α_m are chosen in order to minimize the approximation error when the channel coefficients are approximated with integers. For any given $\mathbf{h}_m$ and $\mathbf{a}_m$ (2.9) is maximized when $\alpha_m = \alpha_m^{MMSE}$, [48] where

$$\alpha_m^{\mathrm{MMSE}} = \frac{\mathbf{h}_m^*\mathbf{a}_m}{1/\mathrm{P} + \|\mathbf{h}_m\|^2}. \tag{2.10}$$

Therefore,

$$\alpha \mathbf{y}_m = \sum_{i=1}^{N} \alpha_m h_{im} \mathbf{x}_i + \alpha_m \mathbf{z}_m = \sum_{i=1}^{N} a_{im} \mathbf{x}_i + \underbrace{\sum_{i=1}^{N} \left(\alpha_m h_{im} - a_{im}\right) \mathbf{x}_i + \alpha_m \mathbf{z}_m}_{\text{effective noise}} \tag{2.11}$$

The effective SNR at the receiver of relay m becomes

$$\mathrm{SNR}_m = \frac{\mathrm{P}}{|\alpha_m|^2 + \mathrm{P} \left\| \alpha_m \mathbf{h}_m - \mathbf{a}_m \right\|}. \tag{2.12}$$

From (2.12) it is obvious that in addition to the channel noise, the CF scheme introduces another source of noise due to integer approximation of non-integer channel coefficients.

In the original CF scheme, upon recovery, the linear combinations are forwarded to the centralized decoder that recovers $\hat{\mathbf{x}}_i$ from $\mathbf{u}_m$. The recoverability condition is given as follows:

$$rank\left(\left[\Re\left(\mathbf{A}\right)\right]_q\right) = rank\left(\left[\Im\left(\mathbf{A}\right)\right]_q\right) = N \tag{2.13}$$

over $\mathbb{F}_q$, where $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N]$.

In [7] the CF scheme is compared with other relaying strategies for TWRN. It is shown that the CF scheme significantly outperforms other relaying strategies in terms of the rate i.e. bits per channel use, and approaches the upper bound with the increase of SNR.

Nazer and Gastpar's original approach [48] requires the codewords to be from a large finite field $\mathbb{F}_q$ with prime q. However, practical communication systems typically employ constellations with sizes equal to powers of two. When q is not prime, or small (e.g. $q = 2^p$), the necessary and sufficient conditions for recovery of $\mathbf{x}_A$ and $\mathbf{x}_B$ with respect to the integer coefficients [48, Th. 8] are often not satisfied. Therefore, a recent work [49] further develops a CF scheme suitable for practical communication systems. In addition, this work provides an example of code design for practical CF or integer-forcing receivers based on the LDPC codes.

A modification of the CF scheme utilizing phase precoding at the terminals is introduced and investigated in [50]. The authors show that phase precoding increases the computation rate compared to the original CF scheme. As a result, the coding gain of 4 dB was achieved at the equation error rate of 10^{-4}. In spite of the obvious benefits,

because the implementation of precoders significantly increases the complexity of communication systems, many practical solutions avoid using phase precoders. In addition, phase precoding is not so beneficial in fast-fading channels.

In [51] Hern and Narayanan propose a multilevel coding scheme for CF which does not require the code used to be a lattice code, but only requires the code to be linear. As a result, the proposed CF can be implemented with lower encoding and decoding complexity. At the same time, it allows error correction for a larger class of decoding functions compared to the original CF scheme. Another work [52] analyzes the degrees of freedom of the CF system composed of K transmitters and K relays. It is demonstrated that the lattice implementation of CF relaying only achieves degrees of freedom less than or equal to 2. Subsequently, the authors proposed a new CF implementation achieving K degrees of freedom. Other constructions of practical CF codes are proposed in [53–56]. The extension of CF with MIMO is investigated in [57].

2.4 Synchronization in PNC

The synchronization requirement in PNC means that terminals A and B should be synchronized in terms of their time and frequency. Some early PNC schemes also required phase synchronization i.e. zero phase difference between the terminals. However, in this literature review we consider the phase difference as a matter of a mapping algorithm. Accordingly, the presence of a frequency offset causes a relative rotation of one transmitted signal around the other. As a result, the phase difference is constantly being changed within the packet duration [17]. If the frequency offset is accurately estimated, the relative phase difference can be reflected in the mapping algorithm, e.g. the tracking of the channel coefficients. The methods of channel estimation in the presence of frequency asynchrony are well-developed for MIMO systems. Therefore, our main focus is limited to time synchronization. We also note that tight synchronization between the relay and a terminal is usually not required.

Synchronous arrival of two packets to the relay is a crucial requirement for any PNC scheme, with the exception of those based on ANC. Figure 2.2 illustrates an example of

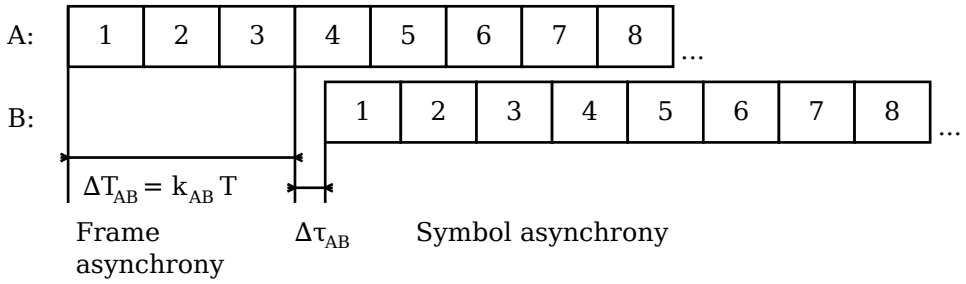


Figure 2.2: Symbol and frame asynchrony. The integer number of misaligned symbols k_{AB} represents frame asynchrony, and the fractional number of misaligned symbols $\Delta\tau_{AB}$ represents symbol asynchrony.

misalignment of two packets due to lack of synchronization. Without loss of generality, we assume that packet $\mathbf{m}_A$ arrives earlier than $\mathbf{m}_B$. Let $\Delta t_{AB} > 0$ be the time difference between $\mathbf{m}_B$ and $\mathbf{m}_A$ arrivals, such that

$$\Delta t_{AB} = k_{AB}T + \Delta\tau_{AB}, \tag{2.14}$$

where k_{AB} is the integer number of symbol periods T, and $0 \leqslant \Delta\tau_{AB} < T$ is the fractional part. Then, k_{AB} represents a measure of frame asynchrony, and $\Delta\tau_{AB}$ quantifies symbol asynchrony.

Most studies on PNC assume perfect symbol synchronization. Other works investigate the performance of PNC in the presence of symbol asynchrony and demonstrate that the lack of synchronization causes severe performance degradation [58]. However,

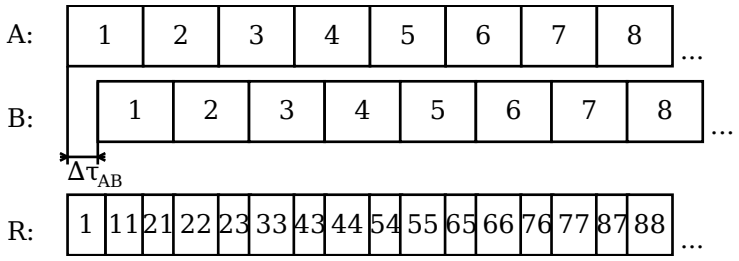


Figure 2.3: A technique for PNC decoding in presence of symbol asynchrony. Many asynchronous PNC methods assume that the symbol offset is known, and the relay can sample the input signal with the sampling period less than the offset, i.e. $T_s < \Delta\tau_{AB}$.

a number of recent theoretical studies [59–63] propose PNC schemes robust for symbol asynchrony. The common method of analysis in these works is illustrated in Figure 2.3. According to this method, the relay is able to reliably estimate time offset between two packets arriving simultaneously but not synchronously. Then, the receiver samples the superimposed signal with doubled sampling rate, and furthermore, such sampling is non-uniform. Accordingly, the desired signal is recovered by applying various techniques. For example, the results obtained in [59] suggest that asynchronous PNC yields satisfactory performance when belief-propagation decoding methods are applied. Another concept with the aim of combating the impact of symbol asynchrony is proposed in [60] for unchannel-coded PNC and in [59] for channel-coded PNC. These papers suggest a concept of network coding [60] or joint channel decoding and network coding [59] based on belief propagation. The results of these papers show that for unchannel-coded PNC, belief propagation can significantly reduce the asynchrony penalties compared with other methods. Furthermore, when the channel coding is presented in PNC, the presence of asynchrony improves the system BER performance, compared to the case when two transmitters are fully synchronized. The conclusion of [59] suggests that symbol asynchrony is no longer a major concerns in PNC if channel coding is utilized.

However, the implementation of such methods, imposes some difficulties. For instance, doubling the sampling rate in a software receiver may not always be desirable, as it causes higher computational delay. Finally, decoding algorithms for asynchronous PNC are often based on the assumption that the pulse-shaping function is rectangular. This assumption allows the output of the matched filter to be sampled at any time within $\Delta\tau_{AB}$. In practice, however, the pulse-shaping function takes a different form, for example the raised-cosine shape. As a result, the proposed methods should be further studied before they can be applied.

Perfect frame synchronization between two transmitters is also hard to achieve. However, [32] and [64] show that a minor lack of frame synchronization is a benefit rather than a problem, and should be exploited in PNC system design. One such benefit is that unaligned interference-free symbols in the beginning and the end of the received signal can be used for channel estimation. In addition, either a prefix or suffix added to a packet is

in the interference-free part of the received superimposed signal. Therefore, it allows the estimation and subsequent compensation of the frame offset.

Recently, a number of PNC architectures which are initially designed to be reliable against the lack of synchronization have been proposed. One of them, OFDM-PNC, presented in [65], is based on the fact that in OFDM, a data stream is represented by a number of lower-rate streams in different subcarriers. As the symbol duration in each subcarrier is larger than that in the time domain, the use of OFDM improves symbol alignment in each subcarrier. Therefore, PNC mapping, performed independently in each subcarrier, suffers less from asynchrony, making the overall OFDM PNC system more robust.

2.5 Channel estimation in PNC

Most studies on PNC assume that the relay is aware of the channel state information i.e. the channel coefficients h_{AR} and h_{BR} are known. In fact, however, the relay should be able to estimate channel coefficients at the beginning of each packet, based on the superimposed received signal.

When the symbol synchronization error τ_{AB} is small compared to the symbol period T, the problem of PNC channel estimation can be seen as a MIMO or distributed MIMO channel estimation problem. As a result, a number of well-developed MIMO channel estimation techniques can be applied [67]. For example, according to the least squares training-based channel estimation method, training vectors $\mathbf{t}_1, \ldots, \mathbf{t}_N$ of length L, known at the relay, where N is a number of terminals transmitting simultaneously, are transmitted in front of the respective packet $\mathbf{x}_1, \ldots, \mathbf{x}_N$. Therefore, the relay receives

$$\mathbf{r} = \mathbf{h}\mathbf{T} + \mathbf{z}, \tag{2.15}$$

where $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$ is $L \times N$ training matrix, and $\mathbf{z}$ is the channel noise. Applying the least squares method, the channel state is estimated as follows:

$$\hat{\mathbf{h}}_{LS} = \mathbf{r}\mathbf{T}^{\dagger}, \tag{2.16}$$

with $\mathbf{T}^{\dagger} = \mathbf{T}^{*} \left(\mathbf{T}\mathbf{T}^{*}\right)^{-1}$. In addition, the work shows that the channel estimation error is minimized if the training vectors $\mathbf{t}_1, \ldots, \mathbf{t}_N$ are orthogonal and of the same norm. Furthermore, [67] and references therein address some alternative methods of MIMO channel estimation. At the same time, the impact of small symbol asynchrony on the performance of distributed MIMO systems is investigated in [68].

In general, however, τ_{AB} is comparable with the symbol duration T. In this case the channel estimation methods described in [69,70] can be applied.

On the other hand, a number of PNC schemes do not require knowledge of the channel state information. For example, [71] proposes an ANC scheme with differential modulation (ANC-DM). The simulation results show that the BER performance penalty of the scheme is about 3 dB compared to a similar coherent ANC scheme. Furthermore, since the proposed scheme is based on the AF relaying, it demonstrates the common drawbacks of ANC, such as amplification of noise and the difficulty of integration of this scheme with channel coding techniques.

2.6 Network coding prototyping efforts

Prototyping of a real-time network coding testbed requires the implementation of a network coding scheme and a relevant ARQ protocol, and subsequently, their integration. Recently, a number of works have reported the successful implementation of DNC testbeds which yield a significant increase of the network throughput compared to the traditional approaches [18–22,72,73]. In contrast, only a few results are available on the implementation of PNC, despite the fact that the theory of PNC has been quickly developing in the last decade. In [32] the authors describe an implementation of analog PNC (ANC). In their experiments, the ANC scheme increased the network throughput by up to 70% compared to traditional scheduling methods. The difference between this result and the theoretical doubling of the throughput was explained by the lack of frame synchronization between the terminals. The performance also deteriorated from the inherent disadvantage of the ANC scheme, i.e., that the relay simply amplifies the noisy version of a signal it receives, without denoising it, and forwards the signal, hence propagating the

noise. A more recent paper ([74]) presents the implementation of a TWRN testbed with the OFDM PNC scheme. In this scheme, the PNC mapping is performed independently across each subcarrier. The fact that the symbol duration in each subcarrier in the OFDM is larger than that in the time domain increases the robustness of the system against symbol asynchrony. In a very recent work [75] a prototype of TWRN with asynchronous PNC is presented. In this testbed, the relay performs well against symbol asynchrony, however, it often experiences the situation when two packets are frame-asynchronous. In this case, the relay drops the packets and this results in degradation of the actual network throughput.

To the best of our knowledge, reports on the successful prototyping of the CF relaying scheme have not been published before our work [76].

2.7 Software-defined radio

Software-defined radio (SDR) is a radio system in which all signal processing components are implemented entirely in software rather than in analog circuits. Reprogramming software, if needed, can be done much faster and cheaper than the replacement of analog circuits. As a result, SDR presents extreme flexibility and programmability compared with traditional communication systems. Therefore, it appears to be a good basis for fast prototyping of communication systems. The reader is referred to [77–80] for more details on the history of development of the SDR concept.

Figure 2.4 illustrates two of the most common architectures of an SDR receiver. In the first architecture, shown in Figure 2.4(a), the intermediate frequency (IF) signal processing of the superheterodyne is implemented as a part of the analog RF circuit, but the baseband signal processing is performed digitally. The second architecture presented in Figure 2.4(b) allows further shrinking of the analog part, by supporting the IF signal processing in a digital manner, on a high-performance FPGA. As a result, the second architecture is more flexible compared to the first. In addition to these two schemes, the third architecture, so-called *ideal software radio* is theoretically possible. In this case, the ADC and DAC are connected directly to the antenna, and the rest of signal processing is

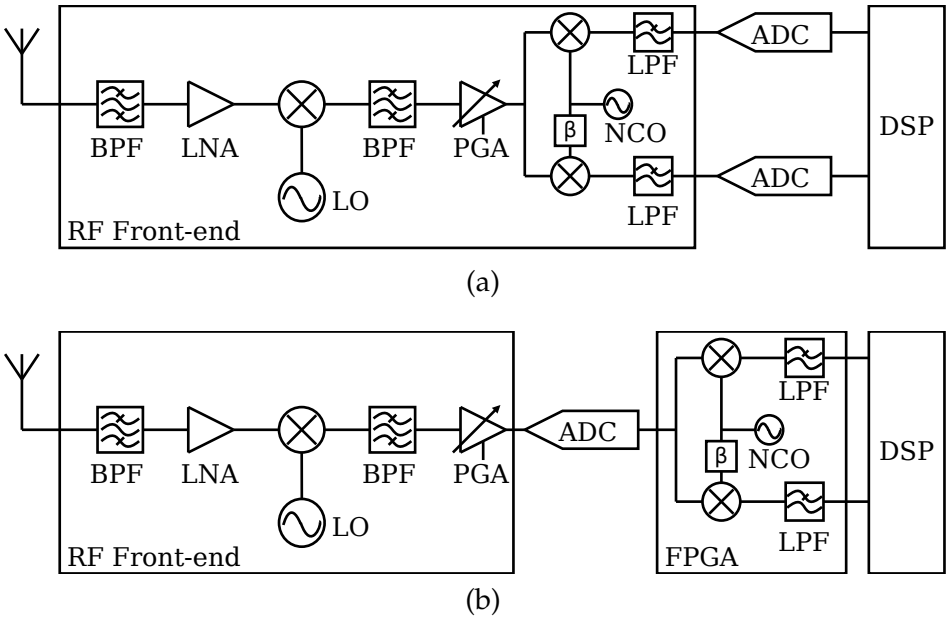


Figure 2.4: Two SDR receiver architectures. The superheterodyne, (a) with analog IF signal processing and (b) with digital IF signal processing, performed by an FPGA.

performed digitally. This concept is implementable at present, especially for VHF/UHF communications, as multi Gigasamples per second ADCs and DACs are available. However, the use of such converters is often impractical due to their high cost, low effective resolution [81], and redundancy of such sampling. Therefore, the second SDR architecture is often preferred. In particular, it is implemented on all USRP devices including USRP N210.

Since SDR represents a perfect environment for the development and prototyping of new-generation communication systems, it is gaining more popularity. For example, a number of testbeds prototyping the emerging wireless communication technologies, such as massive MIMO [82–84], LTE [85], full-duplex transceivers [86] and mobile phone base station, OpenBTS [87] have been implemented entirely on SDR platforms.

Due to the growing popularity of SDR, there is a strong demand for a universal and standardized development platform providing compatibility of different modules from different projects, something like "SDR Matlab". Such a platform should also provide a rich library of ready-made modules. Today several such SDR platforms are available, including commercial platforms such as NI LabView, Matlab Simulink and Nutaq SDR

(former Lyrtech), as well as open-source platforms e.g. GNU Radio, SODA [88,89], SORA [90], CalRadio [91], REDHAWK SDR Framework [92] and the WARP project [93], to name only a few. Of these, GNU Radio appears to be one of the most popular due to its availability and flexibility achieved with the fact that its source codes are open. The full availability of all source codes allows users to take advantage of crowdsourcing, when many independent users can test the blocks and contribute to GNU Radio development. Furthermore, with the source code open, individual users can modify the algorithms implemented in a block depending on their particular demands. In contrast, a modification of the signal processing algorithms in a block of a commercial SDR system is impossible, because the source codes are typically not available.

Despite the obvious benefits, a number of disadvantages are typical for all SDR systems. First, since the signal processing is done using a regular processor instead of dedicated hardware, what can be done is limited. Second, many SDR systems, including GNU Radio and NI Labview, are not clocked. Instead, the way each block is given access to the CPU resource is decided by a scheduler. Although special algorithms of the scheduler attempt to minimize the waiting time, it is still larger than that of clocked systems. Therefore, SDR systems demonstrate the increased random delay of signal processing. Finally, many SDR libraries mainly address signal processing in the physical layer, without consideration of the upper layers [94]. Therefore, the libraries for the layers, other than the physical layer, are often under-developed, and sometimes support only outdated algorithms.

2.7.1 GNU Radio

GNU Radio is a free open-source software development toolkit that provides the environment and libraries of blocks for the implementation of software radios using external low-cost RF hardware, such as Ettus USRP. GNU Radio allows users to implement real-time, high-throughput radio systems in a simple-to-use, rapid-application-development environment [95].

The architecture of a GNU Radio project, also referred to as a *flow graph*, is sketched in Figure 2.5. A flow graph consists of a number of connected blocks. Typically, there

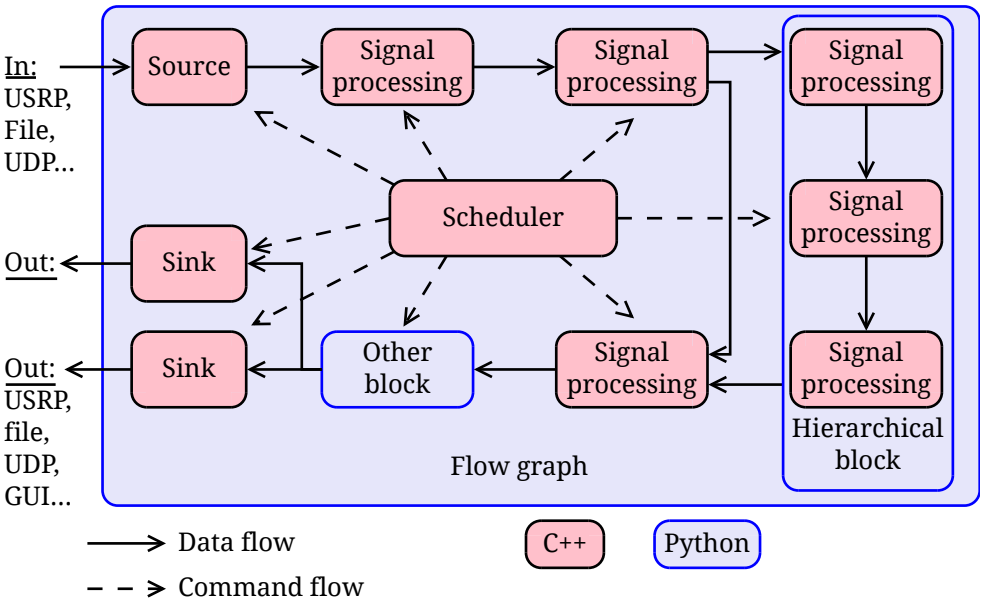


Figure 2.5: GNU Radio flow graph architecture. A flow graph is composed of sources, signal processing blocks and sinks connected to each other. Primitive blocks are implemented in C++, while Python is used for connecting the blocks and constructing the flow graph. The scheduler controls the data flow and calls the blocks to process data.

is one or a few source blocks which provide data without consuming other data. Therefore, source blocks have only output ports. The data can either be imported from a file, UDP interface, USRP, or a locally-generated signal such as a waveform. Next, signal processing blocks have both input and output ports. There are a wide range of processing blocks available, including simple arithmetic blocks, such as addition and multiplication, and complex communication signal processing blocks such as filters, modulators and synchronizers. Finally, sink blocks have only input ports and are used to output the processed data out of the flow graph. Examples of sink blocks are USRP sinks, audio sinks connected to the sound card, file sinks or visualization blocks. The flow graph is written in Python, while blocks are either written in C++ or Python. Typically, signal processing blocks are implemented in C++, while other non-signal-processing blocks, such as valves or selectors, may be implemented in Python. In addition, there are so-called *hierarchical blocks*, the superblocks, composed of a few smaller blocks with the connection provided in Python.

A flow graph is controlled by the GNU Radio *scheduler*, which allocates computa-

tional resources of the CPU to the blocks and determines the order for blocks to be called. The scheduler attempts to arrange the signal processing in blocks in such a way to minimize the overall latency of the flow graph and maximize the signal flow. The behavior of the scheduler is explained in detail in [96]. By default, the scheduler prefers to process continuous data streams. Therefore, if a block does not produce the output continuously, the scheduler may call the block several times repeatedly without a result, rather than giving the hardware resource to other blocks. This situation may increase the delay of the system. However, the new design of the scheduler introduced in GNU Radio 3.7 and subsequent versions provides better options for tuning the scheduler. In this way, the user can define the maximum and minimum output buffer size of a block, giving the scheduler a clue about the expected behavior of the block. At the same time, modification or tuning of the scheduler requires substantial experience, as wrong settings can ruin the performance of a flow graph.

When a flow graph processes data from a real-time hardware such as a USRP, it must provide real-time processing of the data. If the source supplies data samples too fast, so that the scheduler cannot ensure real-time processing, this exception is referred to as *overflow*. The opposite situation, when the flow graph fails to provide continuous flow of output samples with the required sampling rate, is known as *underflow*. Handling overflows and underflows breaks the normal order of the scheduler's work and should be avoided as far as possible.

GNU Radio latency for the case when GNU Radio is used together with Ettus USRP is analyzed thoroughly in [97]. The paper identifies the sources of latency and estimates their contribution to the total latency of the platform. According to the paper, the total latency is still high enough for the implementation of current wireless network protocols. Hence, at present, GNU Radio seems to be more suitable for the prototyping and evaluation of emerging wireless communication algorithms in the research environment, rather than for replacing the traditional hardware-based communication systems. In contrast, a recent work [98] describes SDR implementation of a Wi-Fi receiver, the performance of which is comparable to that of a consumer-grade hardware Wi-Fi card. This work shows that many problems of GNU Radio can be avoided if the software is designed in the ap-

propriate, computation-efficient way. A useful tool which allows to analyze flowgraphs and identify slow blocks is presented in [99].

In addition, over recent years, the GNU Radio project has been actively developed. As a result, the latency is decreasing in each new version of GNU Radio, compared to the previous versions. This is achieved by optimization of the scheduler [96], vectorization [100], the use of GPU and other methods. Therefore, we believe, that latency will not be a major issue in the next generation GNU Radio.

2.8 Universal software radio peripheral

Universal Software Radio Peripheral (USRP) is a family of hardware products developed and manufactured by Ettus Research. USRPs are designed to be a mid-range price hardware platform for SDR systems, affordable, yet providing high performance and sufficient quality for quick prototyping and research experiments in universities and research labs. Most USRPs are connected to a computer using the USB interface or Ethernet cable. The connection type depends on the specific USRP model. For transmitting, a sampled baseband signal is generated in the computer and sent to the USRP, where it is converted to an analog signal and up-converted onto an RF carrier. In receiving mode, it receives an RF signal, downconverts it, digitalizes it and sends it to the computer. In such a way, the USRP can be seen as a high-speed ADC/DAC peripheral for the PC.

A USRP consists of a motherboard with mounted FPGA and ADCs/DACs, and can accommodate one or two replaceable daughterboards. The number of daughterboards depends on the USRP model. Since daughterboards represent the Tx and Rx RF front-end, they can handle analog signals only. Different types of daughterboards are designed for different applications, therefore each type provides a different frequency range. With all the different types of daughterboard available, it is possible to tune in to frequencies from DC to 5.9GHz.

USRP Hardware Driver (UHD) is a driver that controls USRP from the host PC, configures it and sets it up properly. When the UHD is used with GNU Radio, it is accessible

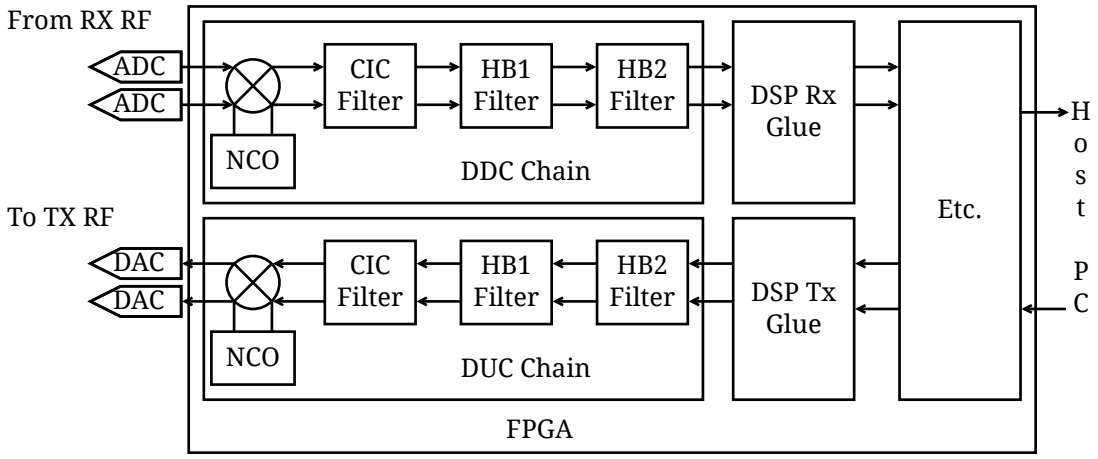


Figure 2.6: DSP components of the standard FPGA image of USRP N210. Modules DSP Tx Glue and DSP Rx Glue are empty by default, but represent a space for user customization of the image. Module named as Etc. represents other non-signal-processing modules omitted in this diagram for simplicity.

via GNU Radio UHD source and sink blocks. USRP devices serve as a primary hardware for GNU Radio, however, the UHD libraries can also be used for the design of standalone applications in C/C++. In addition to UHD and GNU Radio, the use of USRP devices is also supported by Matlab Simulink and NI LabView, which have their own proprietary drivers other than the UHD.

2.8.1 FPGA

Each USRP is equipped with the FPGA that performs the IF signal processing as well as digital down-conversion (DDC) and digital up-conversion (DUC). In particular, the FPGA of USRP N210 is represented by the Spartan 3A-DSP 3400 FPGA manufactured by Xilinx, which is able to process signals with up to 100 MS/s sampling rate in both Tx and Rx directions.

In the standard configuration shown in Figure 2.6, the DSP part of the FPGA is represented by the DUC chain in the Tx and DDC chain in the Rx. The main purpose of the DUC is to upsample the Tx signal from the application sampling rate to the FPGA clock rate. In the same way, the purpose of the DDC is downsampling the received signal from the ADC sampling rate to the required application sampling rate. Typically, the applica-

Table 2.1: Standard USRP N210 FPGA Utilization.

Element	Total	Utilized	%
Number of Slice Flip-Flops	47744	19575	41
Number of 4 input LUTs	47744	30079	63
DSP 48A (Fixed-point multipliers)	126	30	24

tion sampling rate is smaller than the FPGA clock rate due to the bottleneck introduced by the Ethernet interface, which connects the USRP and host PC.

The DDC chain receives the signal downconverted from the carrier frequency to the IF and outputs the baseband signal. It consists of the CORDIC numerically-controlled oscillator (NCO), which generates the IF, followed by the cascade of three filters, namely a Cascaded Integrator-Comb (CIC) downsampling filter, and two half-band (HB) downsampling filters. One advantage of a CIC filter is that it can be implemented without multipliers [101]. Similarly, the input of the DUC chain is the baseband signal, and the output is upsampled and upconverted to the IF signal, fed into the DAC. The DUC chain consists of two HB upsampling filters followed by the CIC upsampling filter and another CORDIC NCO module. After the DDC in the Rx chain and before the DUC in the Tx chain respectively, the modules *DSP Rx Glue* and *DSP Tx Glue* are located. By default, these modules are empty and set up as simple pass-throughs. However, users wishing to customize the FPGA image are encouraged to place their signal processing routines for the baseband signal in these modules.

The standard FPGA image only partially utilizes the available hardware resources of the FPGA. Table 2.1 summarizes the available resources of the FPGA and their utilization of the standard image. It is obvious that the available resources of the FPGA are still significant. These resources can be used for customization of the FPGA image, such as implementation of DSP routines or scheduling schemes for channel multiple access. We also note that the relocation of these routines can effectively reduce the impact of common drawbacks of the GNU Radio platform, such as latency and randomness of delay.

2.9 Conclusion

In this chapter we have provided a literature review on recent studies on PNC, as well as the alternatives to PNC, such as DF relaying schemes. Special attention was focused on the implementability and simplicity of different PNC algorithms. It appears that PNC has received significant research attention in the last ten years. However, many studies cannot be directly applied to practice, either because they are based on non-realistic assumptions or their computational complexity is high. In addition, the implementation of some algorithms, which are well-studied in theory, may be impractical, for instance analog network coding. In contrast, other algorithms are specially designed to address a particular implementation challenge such as asynchrony. However, those algorithms are often too specialized, making it impossible to apply them to a slightly changed scenario. For these reasons, only a few simplified testbeds with PNC have been implemented to date. Furthermore, no implementations of the CF relaying strategy have been reported. From this overview it is clear that a gap between theory and practice still exists. Therefore, further efforts are required to make the theoretical results yield practically.

From the implementation point of view, SDR appears to be a suitable environment for prototyping for several reasons. First, SDR provides the utmost flexibility compared to traditional hardware-based development platforms. This flexibility is especially valuable for the testing, debugging and upgrading of a communication system. Second, SDR supports the portability and compatibility of modules from third-party projects. In addition, GNU Radio software together with USRP hardware presents a popular SDR platform where the flexibility can be achieved because all the source codes are open.

Chapter 3

Practical CF scheme design and analysis

The original concept of compute-and-forward (CF) demonstrates superiority over other physical-layer network coding schemes in terms of its throughput performance as well as its efficiency of implementation. However, it is still based on a number of assumptions which hinder its deployment in conventional wireless communication systems. For example, the original CF scheme relies on the use of constellations of prime size q, while the existing communication protocols typically utilize constellations the size of which is a power of two. In this chapter we develop a practical CF scheme for two-way relay network (TWRN) based on the conventional QPSK modulation and Reed-Solomon ECC codes. Since both these components are usually included in standard libraries of existing SDR platforms, including GNU Radio, the prototyping is simplified by eliminating the need to develop custom modules from scratch. At the same time our CF scheme ensures the recoverability of desired packets from the received linear combination at the terminals, which is not always possible in the original scheme when q is small. As a result, with the modified CF relaying scheme, TWRN achieves the throughput predicted by theoretical analysis.

3.1 Introduction

THE compute-and-forward scheme proposed by Nazer and Gastpar in [48] offers a number of advantages compared to other PNC algorithms. At the same time, it overcomes several shortcomings of early PNC schemes, for example, the need for phase alignment between the terminals. In addition, in the context of CF, existing channel coding algorithms can be naturally integrated with network coding, thus increasing the re-

liability of data exchange. Moreover, the CF scheme can be easily deployed in multi-terminal, multi-relay wireless networks.

On the other hand, in the early works, the CF scheme was mainly analyzed from the information-theoretic point of view. As a result, a number of crucial concerns regarding its employability in actual wireless communication systems were omitted from the scope of those works. One such problem is that the original Nazer and Gastpar approach [48] requires the codewords to be from a large finite field $\mathbb{F}_q$ with prime q. However, practical communication systems typically employ constellations with sizes equal to powers of two. When q is not prime (e.g. $q = 2^p$) or small, the necessary and sufficient condition for successful recovery of the target signal from the linear combinations at the destination decoder [48, Th. 8] is often not satisfied. Specifically, the matrix of linear coefficients is often non-invertible over $\mathbb{F}_q$, when the linear coefficients are optimal with respect to current channel realization. A failure in the recovery results in an extra retransmission, thus causing the network throughput degradation. In order to overcome this, a recent work [49] modifies the original CF scheme to make it suitable for practical communication systems. In particular, in this scheme the linear combinations are estimated over $\mathbb{R}$ rather than over $\mathbb{F}_q$ thus avoiding the case when the matrix of linear coefficients is not invertible over $\mathbb{F}_q$. However, the work assumes that the channel is real-valued.

In this chapter we develop a CF scheme suitable for implementation in GNU Radio. The main concept of our design is that the proposed CF scheme makes use of standard PHY blocks from GNU Radio and Python libraries wherever possible, including ECC and modulation/demodulation modules. In particular, the Python module of Reed-Solomon ECC [102] is utilized. Also, we further extend the CF method of estimating the linear combinations proposed in [49], to make it suitable for use in complex-valued channels with TWRN rather than Gaussian networks. We show that, even with high Tx power, the original CF scheme, when directly applied in our implementation, does not achieve the promised throughput due to the non-invertibility problem. In contrast, with the linear combinations invertible over $\mathbb{C}$ rather than over $\mathbb{F}_2$, the throughput of the TWRN can be increased by about 50% compared to that of the original Nazer and Gastpar approach, thus achieving the theoretical throughput. In addition, we compare the performance of

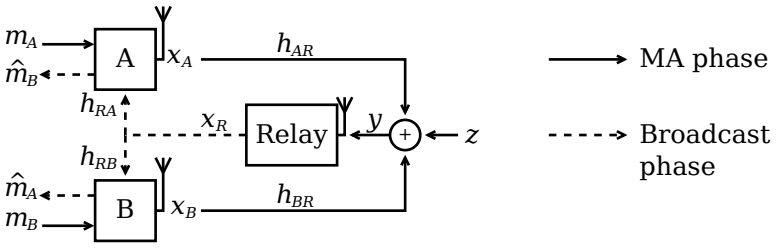


Figure 3.1: Diagram of CF TWRN model. During the MA phase terminals A and B simultaneously transmit packets $\mathbf{m}_A$ and $\mathbf{m}_B$ encoded as $\mathbf{x}_A$ and $\mathbf{x}_B$ with identical codebook $\mathcal{C}$. The relay receives superimposed signal $\mathbf{y}$ and attempts to recover a linear combination $\mathbf{x}_R$. The relay then broadcasts the linear combination during the broadcast phase. Subsequently, terminal A subtracts its own transmitted signal $\mathbf{x}_A$ from $\mathbf{x}_R$ and demodulates $\hat{\mathbf{m}}_B$. Terminal B recovers $\hat{\mathbf{m}}_A$ in the same way.

the designed CF scheme with other non-PNC relaying strategies such as DNC and DF.

The rest of this chapter is organized as follows. First, we introduce the system model of CF two-way relaying in Section 3.2. Then, Section 3.3 describes the construction of the CF encoder using only standard blocks of GNU Radio platform. Next, several ways of finding optimal linear coefficients with respect to channel conditions are presented in Section 3.4. Section 3.5 and Section 3.6 describe the multiple-access (MA) and broadcast phases, respectively, in more detail. In Section 3.7 we provide simulation results, investigate the performance of the proposed scheme and compare it with other relaying strategies. Finally, the summary of this chapter and conclusions are presented in Section 3.8.

3.2 System model

In the CF scheme for TWRN, shown in Figure 3.1, two terminals A and B simultaneously transmit packets $\mathbf{m}_A$ and $\mathbf{m}_B$, each consisting of $2K$ bytes, i.e. $\mathbf{m}_A, \mathbf{m}_B \in \mathbb{F}_{2^8}^{2K}$. Each terminal encodes the transmitted packet with the same lattice codebook $\mathcal{C}$ of length n such that

$$\begin{aligned} \mathbf{x}_A &= \mathcal{C}(\mathbf{m}_A), \\ \mathbf{x}_B &= \mathcal{C}(\mathbf{m}_B). \end{aligned} \tag{3.1}$$

The single antenna relay R receives the following signal

$$\mathbf{y} = h_{AR}\mathbf{x}_A + h_{BR}\mathbf{x}_B + \mathbf{z} \tag{3.2}$$

where $h_{AR}, h_{BR} \in \mathbb{C}$ are the MA phase channel coefficients and $\mathbf{z} \sim \mathcal{CN}(0, \mathbf{I}_n)$ is the complex Gaussian noise. Rather than recovering each packet individually, the relay attempts to recover an integer linear combination of the transmitted signals,

$$\mathbf{x}_R = a_A\mathbf{x}_A + a_B\mathbf{x}_B, \tag{3.3}$$

where $a_A, a_B \in \mathbb{Z}[i]$. Let P be the power of the transmitter, and $\mathbf{a} = [a_A, a_B]^T$.

In the second time slot, the broadcast phase, the relay broadcasts $\mathbf{x}_R$ together with the linear coefficients a_A and a_B embedded in the preamble of the packet. Therefore, terminals A and B receive respectively

$$\begin{aligned} \mathbf{y}_A &= h_{RA}\left(a_A\mathbf{x}_A + a_B\mathbf{x}_B\right) + \mathbf{z}_{RA}, \\ \mathbf{y}_B &= h_{RB}\left(a_A\mathbf{x}_A + a_B\mathbf{x}_B\right) + \mathbf{z}_{RB}, \end{aligned}$$

where $h_{RA}, h_{RB} \in \mathbb{C}$ are the broadcast phase channel coefficients and $\mathbf{z}_{RA}, \mathbf{z}_{RB} \sim \mathcal{CN}(0, \mathbf{I}_n)$ represent noise. Upon reception and channel compensation, each terminal subtracts its own transmitted signal from the received linear combination and demodulates the desired packet $\hat{\mathbf{m}}_A$ or $\hat{\mathbf{m}}_B$.

In this chapter we assume full symbol and frame synchronization between the terminals, and that all channel coefficients h_{AR}, h_{BR}, h_{RA} and h_{RB} are accurately estimated by the respective receiver and remain constant within the packet duration. We also denote $\mathbf{h} = [\hat{h}_{AR}, \hat{h}_{BR}]^T$.

3.3 Codebook construction

We first construct our lattice codebook $\mathcal{C}$ via Construction A, described in [54]. Let $r(x)$ be an irreducible polynomial with a primitive root β, i.e. $r(\beta) = 0$. We let $\mathbb{F}_{2^8} = \dfrac{\mathbb{F}_2(x)}{(r(x))}$,

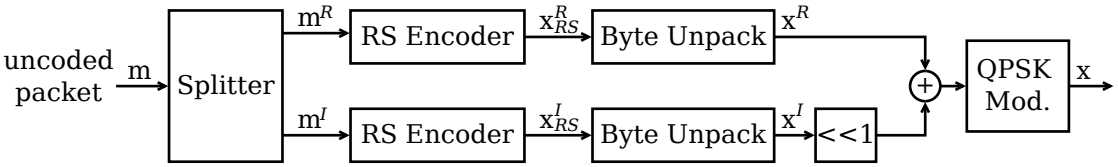


Figure 3.2: Diagram of CF encoder at terminals. Uncoded packet $\mathbf{m}$ is split into two parts $\mathbf{m}^R$ and $\mathbf{m}^I$, and each part is encoded with RS code $\mathcal{C}_{RS}$ and unpacked into bits. These bits $\mathbf{x}^R$ and $\mathbf{x}^I$ are then combined and fed into the QPSK modulator. The output of the QPSK modulator represents codeword $\mathbf{x}$ of lattice codebook $\mathcal{C}$.

where $(r(x))$ is the ideal generated by $r(x)$. In this work we use the Reed-Solomon (RS) error-correcting code $\mathcal{C}_{RS}(N,K)$ over $\mathbb{F}_{2^8}$, where N is the codeword length and K is the packet length. We combine the RS code with QPSK modulation $\mathcal{E}_{\text{QPSK}} : \{0,1,2,3\} \to \{0,1,i,1+i\}$ $(q = 2)$ to construct a lattice codebook $\mathcal{C}$ of length $n = 8N$. Let $\mathcal{C}_{RS}$ be the RS codebook i.e.

$$\mathcal{C}_{RS} = \left\{ \mathbf{c}_{RS} = \mathbf{G}_{RS}\mathbf{w} : \mathbf{w} \in \mathbb{F}_{2^8}^K \right\}, \tag{3.4}$$

where $\mathbf{G}_{RS}$ is the generator matrix of the RS code. Let also $\mathcal{C}_{RS}^U$ be the unpacked RS codebook

$$\mathcal{C}_{RS}^U = \{ \mathbf{x} = \mathcal{U}(\mathbf{c}_{RS}) : \mathbf{c}_{RS} \in \mathcal{C}_{RS} \}, \tag{3.5}$$

where unpacking operator $\mathcal{U} : \mathbb{F}_{2^8} \to \mathbb{F}_2^8$ is the inverse of packing transformation $\mathcal{P} : \mathbb{F}_2^8 \to \mathbb{F}_{2^8}$ with $\mathcal{P}(b_7,\ldots,b_0) = b_7\beta^7 + \cdots + b_0\beta^0$. The codeword $\mathbf{x}$ construction is illustrated in Figure 3.2. First, a stream of uncoded bytes $\mathbf{m} \in \mathbb{F}_{2^8}^{2K}$ is split into two streams $\mathbf{m}^R = [m_1, m_3, \ldots, m_{2k-1}]$ and $\mathbf{m}^I = [m_2, m_4, \ldots, m_{2k}]$, each of them encoded with RS code $\mathcal{C}_{RS}$ to $\mathbf{x}_{RS}^R$ and $\mathbf{x}_{RS}^I$. Subsequently, using $\mathcal{U}$ the streams of bytes are unpacked into streams of bits $\mathbf{x}^R$ and $\mathbf{x}^I$, respectively, and the bits are grouped into chunks of two bits, where the most significant bit is from the first stream $\mathbf{x}^R$, and the least significant bit is from the second stream $\mathbf{x}^I$. These chunks are then fed into the QPSK modulator. This scheme guarantees that the I and Q components of the Grey-coded QPSK constellation are represented by two different RS codewords. Therefore,

$$\mathcal{C} = \left\{ \mathbf{x} = \mathbf{x}^R + i\mathbf{x}^I : \mathbf{x}^R, \mathbf{x}^I \in \mathcal{C}_{RS}^U \right\}. \tag{3.6}$$

Then given $a_A = a_A^R + i\, a_A^I$, $a_B = a_B^R + i\, a_B^I$ and $\mathbf{x}_A, \mathbf{x}_B \in \mathcal{C}$, the received codeword $\mathbf{c}$ at the relay

$$\begin{aligned}
\mathbf{c} &= [a_A \mathbf{x}_A + a_B \mathbf{x}_B]_2 \\
&= \left[\left(a_A^R + i\, a_A^I\right)\left(\mathbf{x}_A^R + i\mathbf{x}_A^I\right) + \left(a_B^R + i\, a_B^I\right)\left(\mathbf{x}_B^R + i\mathbf{x}_B^I\right)\right]_2 \\
&= \left[a_A^R \mathbf{x}_A^R - a_A^I \mathbf{x}_A^I + a_B^R \mathbf{x}_B^R - a_B^I \mathbf{x}_B^I\right]_2 + i\left[a_A^R \mathbf{x}_A^I + a_A^I \mathbf{x}_A^R + a_B^R \mathbf{x}_B^I + a_B^I \mathbf{x}_B^R\right]_2 \\
&= \mathbf{c}^R + i\mathbf{c}^I, \qquad (3.7)
\end{aligned}$$

where $\mathbf{x}_A^R, \mathbf{x}_A^I, \mathbf{x}_B^R, \mathbf{x}_B^I \in \mathcal{C}_{RS}^U$. Since a_A^R, a_A^I, a_B^R and a_B^I are all integers, and $\mathcal{C}_{RS}^U$ is a linear code, $\mathbf{c}^R, \mathbf{c}^I \in \mathcal{C}_{RS}^U$ and therefore $\mathbf{c} \in \mathcal{C}$.

Based on the above construction, the lattice codebook $\mathcal{C}$ has the following property: if $\mathbf{x_A}, \mathbf{x_B} \in \mathcal{C}$, then for all $a_A, a_B \in \mathbb{Z}[i]$, we have

$$\mathbf{c} = [a_A \mathbf{x}_A + a_B \mathbf{x}_B]_q \in \mathcal{C}, \qquad (3.8)$$

where $[\mathbf{v}]_q$ denotes $[v(1) \bmod q, \ldots, v(n) \bmod q]$ for any $\mathbf{v} \in \mathbb{C}^n$.

3.4 Search of optimal linear coefficients

Finding the optimal linear coefficients $\mathbf{a}$ is another practical problem to be solved before the CF scheme is put into practice. The early works suggest an exhaustive search over all coefficients $\mathbf{a}$ and choosing $\mathbf{a}_0$ which maximizes the computation rate (2.9) with given $\mathbf{h}$, as follows

$$\mathbf{a}_0 = \arg\max_{\mathbf{a}} R_{comp}\left(\alpha, \mathrm{P}, \mathbf{a}\right)$$

taking into account that the computation rate is zero if

$$\|\mathbf{a}\|^2 \geqslant 1 + \|\mathbf{h}\|^2 \mathrm{P}.$$

The exhaustive search over this region, however, becomes complicated with the increase of P.

A number of studies [56,103,104] propose more efficient methods for finding the optimal linear coefficients for the CF relaying scheme. However, the authors assume that the channel coefficients are real. This scenario is possible when one dimension constellation such as that of PAM modulation is in use. In our testbed, we make use of a complex constellation, namely QPSK, and consider the possibility to extend it to M-QAM in the future. Therefore, those methods cannot be directly applied in our testbed.

Another approach is based on solving the shortest vector problem. Substituting α_{MMSE} of (2.10) into (2.9) yields

$$R_{comp}\left(\alpha_{\text{MMSE}}, \mathrm{P}, \mathbf{a}\right) = \log^{+}\left(\frac{1}{\mathbf{a}^{*}\mathbf{M}\mathbf{a}}\right), \tag{3.9}$$

where

$$\mathbf{M} = \mathbf{I}_2 - \frac{\mathrm{P}}{1+\mathrm{P}\|\mathbf{h}\|^2}\mathbf{h}\mathbf{h}^{*}. \tag{3.10}$$

To maximize the computation rate (3.9), we should solve the following optimization problem

$$\min_{\mathbf{a}\in\mathbb{Z}^2[i]} \mathbf{a}^{*}\mathbf{M}\mathbf{a}. \tag{3.11}$$

Since $\mathbf{M}$ is a positive definite matrix, it can be decomposed by Cholesky decomposition as $\mathbf{M} = \mathbf{L}\mathbf{L}^{*}$ where $\mathbf{L}$ is a lower triangular matrix. With this, (3.11) can be rewritten as

$$\min_{\mathbf{a}\in\mathbb{Z}^2[i]} \mathbf{a}^{*}\mathbf{L}\mathbf{L}^{*}\mathbf{a} = \min_{\mathbf{a}\in\mathbb{Z}^2[i]} \|\mathbf{L}^{*}\mathbf{a}\|^2. \tag{3.12}$$

The problem in (3.12) is a shortest vector problem for a lattice Λ with generator matrix $\mathbf{L}^{*}$. We employ the complex Lenstra-Lenstra-Lovasz (CLLL) reduction algorithm [105] to solve (3.12). In addition to maximization of the computation rate, we require that both components $\mathbf{x}_A$ and $\mathbf{x}_B$ present in the linear combination $\mathbf{x}_R$, in order to ensure the bidirectional relaying. Therefore, we also require that $a_A \neq 0$ and $a_B \neq 0$.

Other approaches to finding the optimum complex coefficients $\mathbf{a}$ to maximize R_{comp} in (2.9) are also addressed in [50,106,107] and references therein.

3.5 The multiple-access phase

Let both the terminals use the same lattice codebook $\mathcal{C}$. Also, $\mathcal{Z}_{a_A,a_B}$ is a set of all possible sums of constellation points for given a_A, a_B, i.e.

$$\mathcal{Z}_{a_A,a_B} = \{a_A \mathcal{E}_{\text{QPSK}}(a) + a_B \mathcal{E}_{\text{QPSK}}(b) : a, b = 0, \ldots, 2^q - 1\}. \tag{3.13}$$

Using the original CF explained through (3.1) - (3.12), the relay first finds the coefficients a_A, a_B and α, and obtains vector $\mathbf{u}$ by quantizing $\alpha\mathbf{y}$ over $\mathcal{Z}_{a_A,a_B}$

$$u(k) = \arg \min_{z \in \mathcal{Z}_{a_A,a_B}} \|\alpha y(k) - z\|, \tag{3.14}$$

for $k = 1, \ldots, n$. Accordingly, $\left[\mathbf{u}^R\right]_2$ and $\left[\mathbf{u}^I\right]_2$, where $\mathbf{u}^R = \Re(\mathbf{u}), \mathbf{u}^I = \Im(\mathbf{u})$, are separately re-encoded with the code $\mathcal{C}_{RS}^U$ and the relay computes vector $\hat{\mathbf{c}}$

$$\hat{\mathbf{c}} = \mathcal{C}_{RS}^U\left(\mathcal{C}_{RS}^{U\,-1}\left(\left[\mathbf{u}^R\right]_2\right)\right) + i\mathcal{C}_{RS}^U\left(\mathcal{C}_{RS}^{U\,-1}\left(\left[\mathbf{u}^I\right]_2\right)\right). \tag{3.15}$$

Next, using $\hat{\mathbf{c}}$ and a simple slicer [49], the relay estimates $\mathbf{x}_R$ in (3.3) based on the following decision rule

$$x_R(k) = \arg \min_{j \in (\hat{c}(k) + q\mathbb{Z}[i]) \cap \mathcal{Z}_{a_A,a_B}} \|\alpha y(k) - j\|, \tag{3.16}$$

where $k = 1, \ldots, n$. This means, that first the channel noise may cause errors in the quantization procedure (3.14). However, these errors are corrected by the RS code when the relay computes $\hat{\mathbf{c}}$. Assuming that $\hat{\mathbf{c}}$ was correctly decoded, the relay quantizes $\alpha\mathbf{y}$ again, however over a smaller subset of $\mathcal{Z}_{a_A,a_B}$, namely each $\alpha y(k)$ is quantized over $(\hat{c}(k) + q\mathbb{Z}[i]) \cap \mathcal{Z}_{a_A,a_B}$. As the minimum distance between the points of $(\hat{c}(k) + q\mathbb{Z}[i]) \cap \mathcal{Z}_{a_A,a_B}$ is higher than that of $\mathcal{Z}_{a_A,a_B}$, the quantization (3.16) is more robust against channel noise than (3.14). An example of this scheme and the relationship between $\mathbf{x}_R, \hat{\mathbf{c}}$ and $\alpha\mathbf{y}$ at one time instance are illustrated in Figure 3.3. It is obvious that the minimum distance between the points of $\mathcal{Z}_{a_A,a_B}$ is equal to one, while the minimum distance in each subset $(\hat{c}(k) + q\mathbb{Z}[i]) \cap \mathcal{Z}_{a_A,a_B}$ is equal to two.

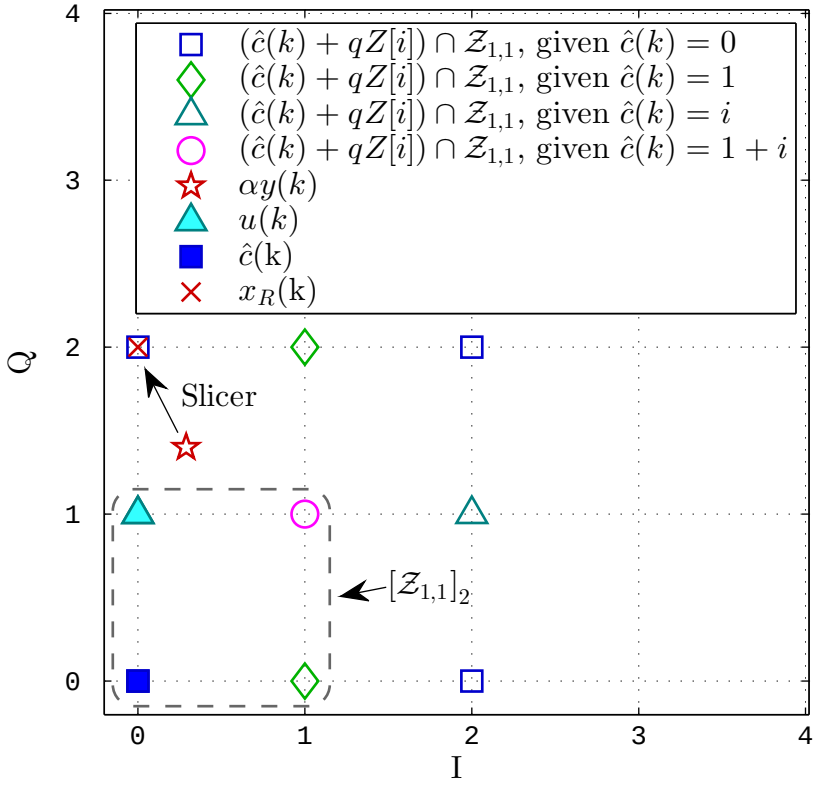


Figure 3.3: An example of estimation of $\mathbf{x}_R$ in (3.16) when $a_A = 1$, $a_B = 1$. Assuming that codeword $\mathbf{c}$ was decoded correctly with the RS code and $\hat{c}_k = 0$, the slicer estimates $x_R(k)$ by quantizing $\alpha y(k)$ only using points from $(0 + 2\mathbb{Z}[i]) \cap \mathcal{Z}_{1,1}$ i.e. $\{0, 2, 2i, 2 + 2i\}$ as quantization levels. As a result, $x_R(k) = 2i$.

Note that before transmission, the constellation points $\{0, 1, i, 1 + i\}$ are centered around zero and normalized in order to transmit the standard QPSK constellation. The centering results in a constant offset of $\alpha y\,(k)$, independent of k. This offset can be estimated and removed accordingly by the relay.

3.6 The broadcast phase

Depending on the MA channel realizations $\mathbf{h}$ and Tx power P which determine the linear coefficients $\mathbf{a}$, the constellation of $\mathbf{x}_R$ may consist of nine points, when $a_A = a_B = 1$, twelve points, when $a_A = 1$ and $a_B = 1 + i$, or sixteen points. Hence, a larger constellation than QPSK is required for the broadcast phase to deliver $\mathbf{x}_R$ to the terminals.

During the broadcast phase, the relay broadcasts the computed signal $\mathbf{x}_R$ mapped

onto the points of 16-QAM constellation, along with coefficients a_A and a_B. Upon reception, terminal A recovers the desired signal by subtracting their own transmitted signal $\mathbf{x}_A$ multiplied by the appropriate coefficient from the noisy version $\hat{\mathbf{x}}_R$ of signal $\mathbf{x}_R$. Let the matrix of channel coefficients

$$\mathbf{U} = \begin{bmatrix} a_A & a_B \\ 1 & 0 \end{bmatrix}. \tag{3.17}$$

Then

$$\begin{bmatrix} \hat{\mathbf{x}}_R \\ \mathbf{x}_A \end{bmatrix} = \mathbf{U} \begin{bmatrix} \mathbf{x}_A \\ \mathbf{x}_B \end{bmatrix} + \begin{bmatrix} \mathbf{z}_{RA} \\ \mathbf{0}_B \end{bmatrix}. \tag{3.18}$$

By choosing $a_A \neq 0, a_B \neq 0$ when searching the linear coefficients, we ensure that matrix $\mathbf{U}$ is invertible over $\mathbb{C}$. Therefore,

$$\hat{\mathbf{x}}_B = \mathbf{v}_1 \begin{bmatrix} \hat{\mathbf{x}}_R \\ \mathbf{x}_A \end{bmatrix} \tag{3.19}$$

where $\mathbf{v}_1$ is the first row of matrix $\mathbf{U}^{-1}$. The target packet $\hat{\mathbf{m}}_B$ becomes

$$\hat{\mathbf{m}}_B = \left[m_{B1}^R, m_{B1}^I, m_{B2}^R, m_{B2}^I, \ldots, m_{BK}^R, m_{BK}^I \right], \tag{3.20}$$

where

$$\begin{aligned} \mathbf{m}_B^R &= \mathcal{P}\left({\mathcal{C}_{RS}^U}^{-1} \left(\Re\left(\hat{\mathbf{x}}_B \right) \right) \right), \\ \mathbf{m}_B^I &= \mathcal{P}\left({\mathcal{C}_{RS}^U}^{-1} \left(\Im\left(\hat{\mathbf{x}}_B \right) \right) \right). \end{aligned} \tag{3.21}$$

Terminal B recover $\hat{\mathbf{x}}_A$ and then $\hat{\mathbf{m}}_A$ in the same way.

3.7 Simulation results

In this section we investigate the performance of the proposed CF scheme with Matlab simulations and compare it with the performance of the DNC scheme and DF scheme

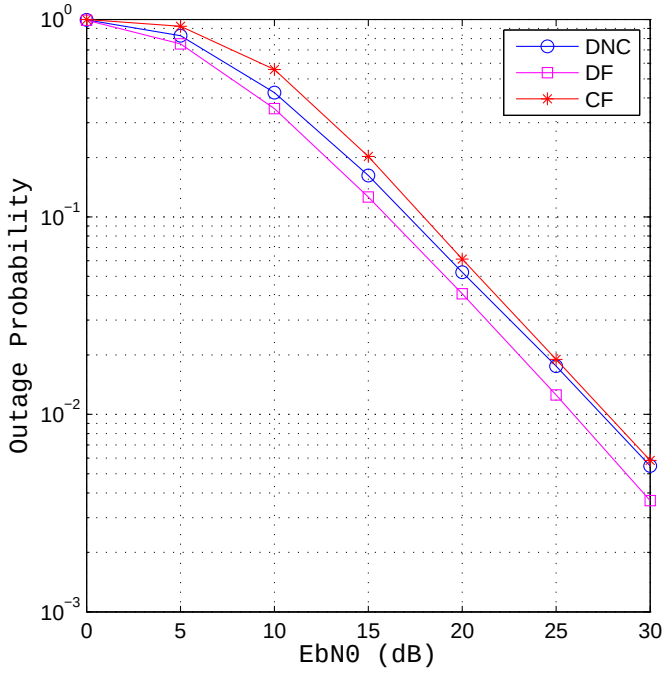


Figure 3.4: Outage probability of the DNC, DF and CF relaying schemes vs. E_b/N_0.

with zero-forcing MIMO receiver, introduced in Section 2.2.1. The channel is a zero-mean fading channel i.e. $h_{AR}, h_{BR}, h_{RA}, h_{RB} \sim \mathcal{CN}(0,1)$. In this section we only provide the simulation results based on the RS$(64,56)$ code, because other RS codes demonstrate similar behavior. More specifically, the code is $\mathcal{C}_{RS}(64,56)$ with the first consecutive root β^{110} over $\mathbb{F}_{2^8} = \frac{\mathbb{F}_2(x)}{(r(x))}$, where $r(x) = 1 + x + x^2 + x^7 + x^8$. The generator polynomial has eight roots and is given as

$$\mathbf{g}(x) = \prod_{i=0}^{7}\left(x - \left(\beta^{110}\right)^i\right) \tag{3.22}$$

We first simulate the outage probability of the MA phase of the CF scheme and compare it with that of the DF and DNC relaying schemes based on the same ECC code. The results are shown in Figure 3.4. The figure shows that the CF scheme has the highest outage probability, while the DF scheme shows the lowest outage probability compared to other schemes. This result is expected, because the CF scheme introduces extra noise

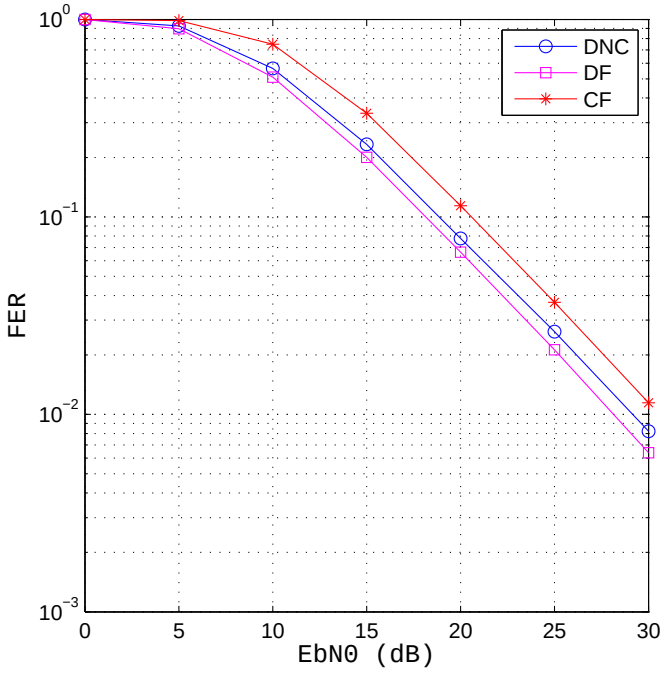


Figure 3.5: User FER of DNC, DF and CF relaying schemes vs. E_b/N_0.

in the receiver due to the approximation of complex channel coefficients by Gaussian integers. Also, unlike the CF, other schemes take advantage of the use of dedicated time slots for transmitting each packet (DNC) or using multiple antennas (DF). However, with high E_b/N_0 the outage probability of the CF scheme becomes similar to that of the DNC strategy.

Next, we consider that both the MA and broadcast phases are noisy, and we investigate the nominal frame error rate (FER) of the TWRN, i.e. when ARQ is not implemented for removing packets with uncorrectable errors. Figure 3.5 compares the FER of TWRN operating with DNC, DF and CF relaying strategies. The results are similar to the previous figure, however, there is a visible gap between FER of the CF and DNC schemes. This gap appears from the fact that the CF relay uses 16-QAM constellation in the broadcast phase, that is larger than QPSK, employed in the broadcast phase of other relaying schemes. It is well known that the BER of 16-QAM is higher than that of QPSK with the same Tx power. As a result, the broadcast phase of the CF scheme contributes more to FER degradation.

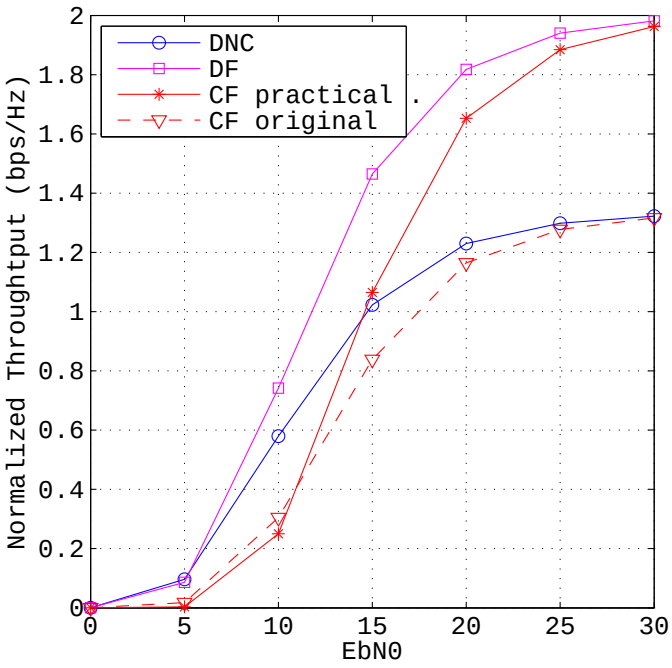


Figure 3.6: Normalized throughput of DNC, DF and two CF schemes vs. E_b/N_0.

We finally consider the throughput of the TWRN. We assume that packets containing ECC uncorrectable errors are detected and discarded by the terminals, for example, with the use of a CRC algorithm. Therefore, those packets do not contribute to the throughput. In this section, we assume the packets do not have an overhead, e.g. address bits or channel estimation sequences, and we define the normalized throughput as the total number of packets N_p of length n, successfully received by both the terminals within time duration t, when t is large:

$$\mathrm{TP} = \lim_{t\to\infty} \frac{N_p \times n \times n_{bps}}{R \times t} \tag{3.23}$$

where R is the symbol rate of the network, and each symbol contains n_{bps} bits. We also assume that no time delay is required for switching between the MA and BC phases. Note that the throughput in a real-time scenario is studied experimentally in Chapter 5. Figure 3.6 compares the normalized throughput of TWRN operating with DNC and DF relaying strategies, as well as our modified CF scheme, all represented by the solid lines. In addition, we plot the throughput of the original CF scheme when it is applied directly,

without modification. The throughput of the original CF scheme is represented by the dashed line. Note that when using QPSK modulation, the possible normalized throughput cannot exceed 2 bps/Hz. We first observe that the original CF scheme does not provide any improvements in terms of its throughput compared to the DNC scheme. This is due to the non-invertibility problem over $\mathbb{F}_2$, when a terminal is unable to recover the target packet from the linear combination, and the terminal therefore requests a retransmission. However, the modified CF scheme does not suffer from the non-invertibility problem. It follows from the figure, that when the E_b/N_0 is higher than 15 dB, the throughput of the modified CF scheme is better than that of the DNC scheme. Furthermore, when E_b/N_0 is more than 25 dB, the throughput of the CF scheme becomes similar to that of the DF relaying strategy, with only a marginal gap. At the same time, CF TWRN does not require the use of a multi-antenna relay. Therefore, despite the higher FER, as illustrated in the previous figure, the CF scheme takes advantage of exchanging packets in two time slots. Yet, with lower energy per bit, even the modified CF scheme provides the lowest throughput, compared to the DF and DNC relaying strategies. Hence, the figure illustrates the advantage of the modified CF relaying strategy when the SNR is high.

3.8 Conclusion

In this chapter we have developed a practical CF scheme that avoids the drawbacks of the original CF scheme, such as the need to use a constellation of large prime size q, that ensures the invertibility of linear combinations in the terminals. At the same time, the proposed CF scheme can be easily implemented on SDR using the standard components typically available in most SDR libraries. We then investigated the performance of the proposed scheme using simulations. The simulation results demonstrate that the DNC scheme outperforms the CF relaying scheme only at low SNR regimes. Despite the fact that the CF scheme has higher outage probability, it takes advantage of two time-slot message exchange, and therefore outperforms the three time-slot DNC scheme in terms of system throughput, when the SNR is medium-to-high.

In this work our practical CF scheme is based on the RS code. However, using the same construction as described in Section 3.3, it is possible to employ other codes, for example LDPC codes. The advantage of the LDPC would be stronger error-correction capability. However, the iterative nature of the LDPC codes would make the synchronization, crucial for the CF scheme, much more complicated and increase the waiting time required to achieve the frame synchronization, thus causing the network throughput to decrease. Furthermore, the computational speed of the LDPC libraries in the current GNU Radio version is more than ten times slower than that of the RS codes. Therefore, currently, the use of the LDPC codes is not a feasible solution on the GNU Radio platform.

In this chapter, we analyzed the performance of the CF scheme, assuming the simplified channel model. As a result, our analysis is incomplete. In fact, in actual channels the performance of the CF scheme may deteriorate due to factors other than channel noise and fading, such as lack of synchronization between the terminals. Therefore, we discuss the impact of implementation issues in Chapter 5 and propose our solutions to minimize them.

Chapter 4

Data-link layer protocols for PNC and their implementation on SDR

In addition to developments in the physical layer, the implementation of a real-time PNC/CF testbed requires developments in the data-link layer algorithms for detecting corrupted packets and acknowledging the transmitter accordingly. However, the algorithms developed for the traditional relay networks or those with DNC relaying cannot be directly applied to PNC networks. In this chapter we present our design of the data-link layer protocol for the CF TWRN and its implementation in GNU Radio. We also compare the performance of the protocol with those utilized in DNC and DF TWRN and discuss the advantages and disadvantages of each protocol.

4.1 Introduction

WHILE most studies on PNC focus on designing novel algorithms for the physical layer, PNC-motivated development for the upper layers has received considerably less attention. However, in order to integrate PNC algorithms with the existing wireless communication systems or to prototype them, developments in the physical layer should be reflected in the upper layers, including the data-link layer. For instance, PNC relays are not as intelligent as DNC or DF relays, because they do not recover the packets bit-by-bit. Therefore, the data verification algorithms in the data-link layer cannot be used directly.

To be put into a real-time prototype, each relaying strategy requires an automatic repeat request (ARQ) protocol that fits into the strategy. Such a protocol is needed for data

integrity verification, and provides a mechanism for the detection of packets damaged beyond the ECC's capability to correct them and missed packets. Subsequently, the ARQ protocol requests for retransmission of those packets. In fact, the performance of an ARQ protocol can be optimized if it is aware of how the relaying is organized in the physical layer. If not optimized, the ARQ protocol may negatively affect the actual performance of a relaying scheme, making it far worse than predicted theoretically. In this chapter, we describe in detail important features of ARQ protocols which we have implemented in our testbed for the DNC, DF and CF relaying strategies and highlight the differences between them.

In addition, those ARQ protocols need to be adapted for their use in GNU Radio. GNU Radio represents a mature SDR platform for developments in the physical layer, but demonstrates lack of maturity in the upper layer, although that is a common problem for many other SDR platforms [94]. Therefore, implementation in GNU Radio requires taking into account special features of this platform, such as the scheduling and block-by-block signal processing, explained in Section 2.7.1. In [73] we presented the design of ARQ protocols for the TS and DNC relaying strategies, specifically designed to cope with GNU Radio inadequacies and hardware imperfections. In this chapter we update the ARQ protocol for the DNC strategy and subsequently develop similar ARQ protocols for the DF and CF relaying schemes.

The remainder of this chapter is organized as follows. First, in Section 4.2 we discuss the challenges of implementation of a half-duplex communication between two nodes (either relay or a terminal) in GNU Radio and design a half-duplex packet switching protocol which minimizes the impact of those challenges. Based on this protocol, we design the ARQ protocols for TWRN with DNC, DF and CF relaying schemes in Section 4.3. Performance comparison of these ARQ protocols in presence of the frame asynchrony is provided in Section 4.4. We finally draw conclusions on the data-link layer protocol developments in Section 4.5.

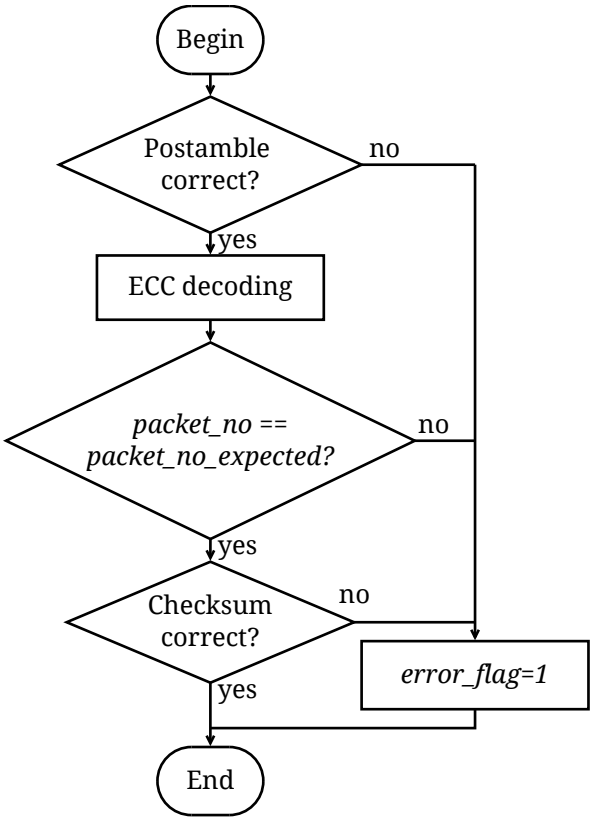


Figure 4.1: Packet verification scheme. When a receiver receives a packet, it first checks if the postamble is correct. Subsequently, the receiver decodes the packet and checks if the packet number matches the expected number. Finally, the checksum of the decoded payload is calculated and compared with that received from the transmitter. If all three conditions are satisfied, the packet is considered to be correct.

4.2 Half-duplex packet switching in GNU Radio

Implementation of TWRN requires regular switching between Tx and Rx, or half-duplex operation. For example, terminal A transmits packet $\mathbf{m}_A$ and switches to listening for the signal, which carries packet $\mathbf{m}_B$. In case of the relay, it receives $\mathbf{m}_A$ and $\mathbf{m}_B$ either sequentially (DNC) or simultaneously (PNC/CF, DF), and relays their XOR or a linear combination, i.e. also constantly switches between Tx and Rx. To simplify the following discussion, we refer to either a terminal or relay as a *node* if a described feature is applicable for both of them.

The current GNU Radio software architecture is primarily aimed at processing continuous data streams. Packet switching is therefore difficult to be implemented in the

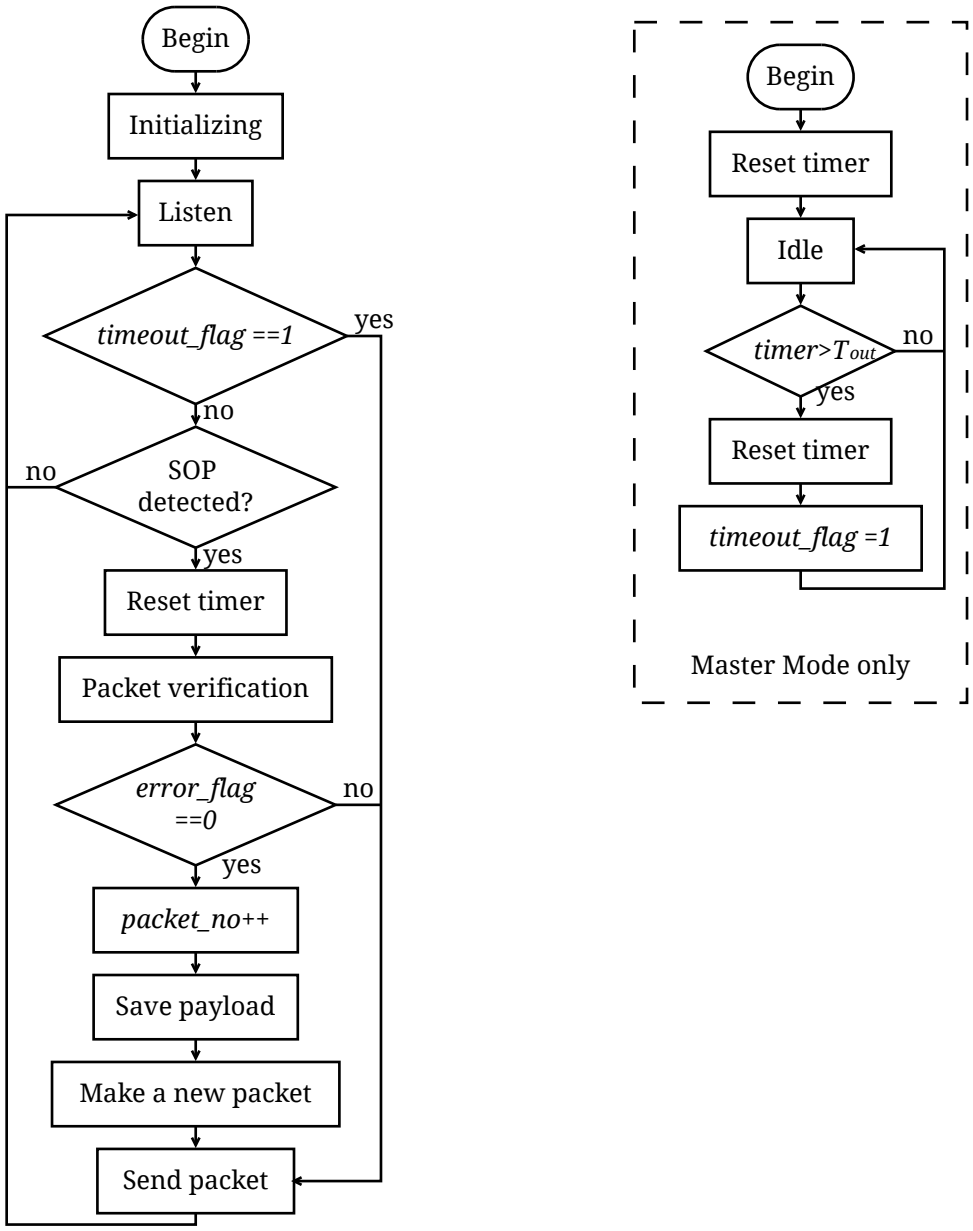


Figure 4.2: A terminal operation flowchart. When a terminal operates in master mode it is equipped with a timer which initiates timeout in case no packets are received within T_{out} time interval from the previous transmission.

GNU Radio receiver, and we have to take this fact into account when designing a half-duplex packet switching protocol. We have chosen the following solutions to minimize the delays occurring due to the non-continuous natute of half-duplex communication:

- First, the acknowledgement of a successful or failed reception is embedded into the

transmitted reply packet, rather than sent individually.

- Second, longer packets are used, as this seems more reasonable from the GNU Radio scheduler optimization perspective.

We employ the transmit-on-receive approach, i.e., a node transmits its packet upon reception from another node. This approach identifies the node starting communication as the *master* and the other as the *slave*. Each packets is equipped with the sequence number which helps to identify any missed packets. When a packet is received by a node, the node receiver verifies its integrity. Figure 4.1 illustrates the utilized packet verification scheme. First, the receiver checks if the tail of the packet is correct. An incorrectly received end of packet usually indicates the presence of an insertion/deletion error [108], underflow at the transmitter, or non-compensated channel rotation during the transmission, i.e., those errors which cannot be recovered with conventional ECC decoding. Thus, the incorrect tail of packet indicates that the receiver should not attempt to decode the damaged packet. Second, after ECC decoding, the packet number is checked. The node compares the packet number of the received packet with the expected packet number. A packet with an unexpected packet sequence number is also considered to be damaged. Third, the checksum of the decoded payload is calculated and compared with the received checksum calculated at the transmitter. Mismatch of these two checksums indicates that there are errors uncorrected after channel decoding. In this testbed we make use of the Adler-32 checksum algorithm from the ZLIB library [109], calculated as follows. Let $\mathbf{d} = [d_1, d_2, \ldots, d_n]$ be a vector of bytes of length n. Then the 32-bit checksum $D_{\text{Adler-32}}(\mathbf{d}) = B \times 65536 + A$, where

$$\begin{aligned} A &= \left[1 + d_1 + d_2 + \ldots + d_n\right] \bmod 65521, \\ B &= \left[(1 + d_1) + (1 + d_1 + d_2) + \ldots + (1 + d_1 + d_2 + \ldots + d_n)\right] \bmod 65521 \\ &= \left[n \times d_1 + (n1) \times d_2 + (n2) \times d_3 + ... + d_n + n\right] \bmod 65521. \end{aligned} \tag{4.1}$$

The preference for the Adler-32 checksum over conventional CRC is determined by the former's higher speed on many platforms [110].

A packet that passes all three steps of verification is considered to be correctly re-

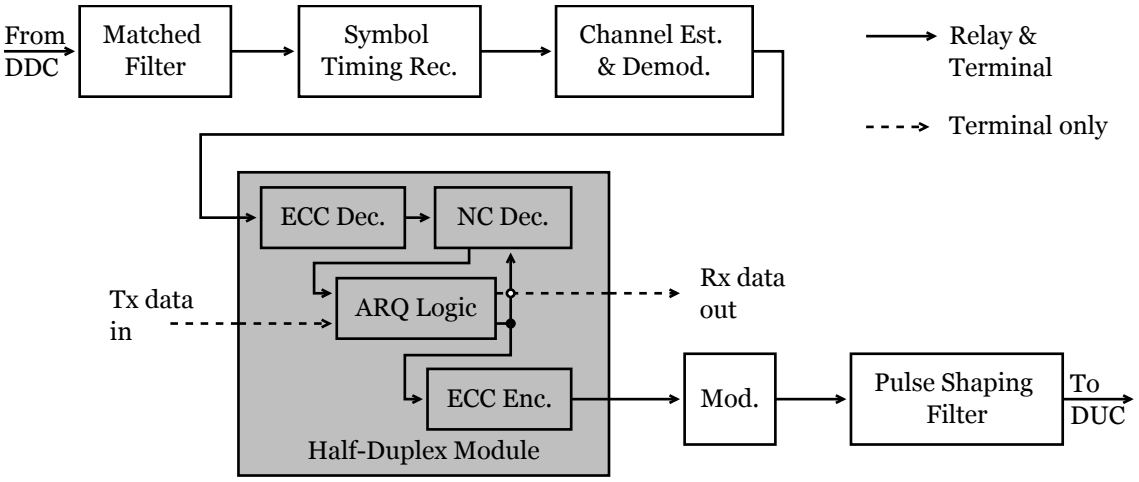


Figure 4.3: Location of the half-duplex block in the simplified block diagram of a node. The half-duplex block is the final block of Rx, and the first block of Tx, thus coordinates Tx and Rx. The white-coloured blocks represent the physical layer blocks processing complex signals, and gray-coloured blocks represent the upper layers blocks processing bytes. Some non-essential modules are omitted.

ceived and its payload is saved or passed to the end-user. Subsequently, the node increments its own packet number and in turn transmits the next packet. A packet that fails verification at any step is marked as corrupted and discarded. Reception of a corrupted packet by a node causes the retransmission of the previous packet, which in turn causes the opposite node to retransmit the packet received corrupted. The full flowchart of this half-duplex packet switching protocol is presented in Figure 4.2. The master retransmits if timeout occurs in case the communication is lost. In contrast, slave nodes never initialize transmitting themselves, but only reply upon receiving a packet from the master.

We implemented this half-duplex packet switching protocol in Python and included it in GNU Radio Companion (GRC) flow graphs as an OOT block, as shown in Figure A.7. The GRC flow graphs and their implementation are described in detail in Appendix A.

Figure 4.3 illustrates the location of the half-duplex block in the simplified block diagram of a node. The half-duplex block is located in the end of the Rx chain, and in the beginning of the Tx chain. It coordinates the operation of the Tx and Rx chains and ensures they do not transmit and receive at the same time. Note that Figure 4.3 shows a simplified scheme. Although the half-duplex blocks in our design are always located on the host PC, different PHY blocks may be implemented either on the PC or FPGA of

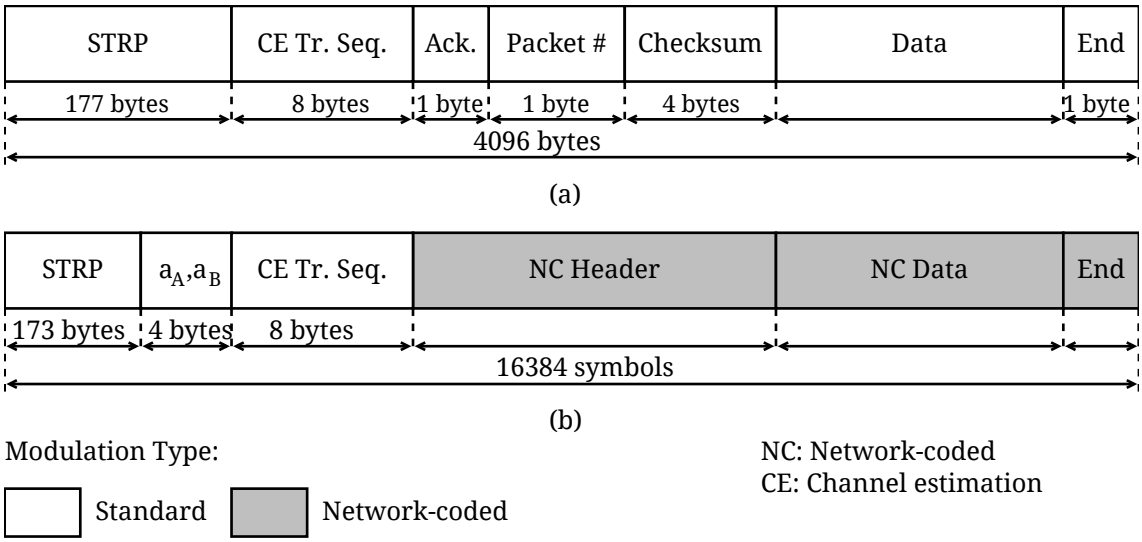


Figure 4.4: Packet format for (a) terminals, DNC relays and DF relays, and (b) CF relays. Both types of packet include symbol-timing synchronization preamble, channel estimation training sequence, header, data and the end of packet symbol. The header consists of acknowledgement, packet number and the checksum. In addition, packets transmitted by CF relays contain the linear coefficient a_A and a_B of the broadcasted linear combination.

USRP. The exact location of those PHY blocks is discussed in Section 5.3.3.

4.2.1 Packet formats for different relaying strategies

Based on the needs described above, we first define the packet format for non-PNC communications, such as those used by the terminals, as well as the DNC and DF relay. This format is sketched in Figure 4.4(a). The length of the packet is 4096 bytes. This means that, if packets are modulated with QPSK, which encodes 2 bits per symbol, 4 symbols per byte, the packet length in symbols is equal to 16384. A packet consists of the preamble, header and encoded payload. A preamble begins with a symbol-timing recovery preamble (STRP), the length of which is 177 bytes. The symbol-timing recovery is used for obtaining the right sampling time before demodulation and is explained in detail in Section 5.4. STRP is followed by the 8 byte training sequence for channel estimation at the receivers. This sequence also represents the address of the intended receiver. The header is composed of one byte for acknowledgement, one byte for the packet sequence

number and four bytes of the checksum. The header is followed by the payload, and both are encoded into 62 codewords with the $(64, 56)$ Reed-Solomon (RS) code. In CF terminals these 62 RS codewords are further processed according to the scheme described in Figure 3.2 in order to construct 31 consecutive codewords from lattice codebook $\mathcal{C}$. The end of packet is represented by one byte with the value $0x55$.

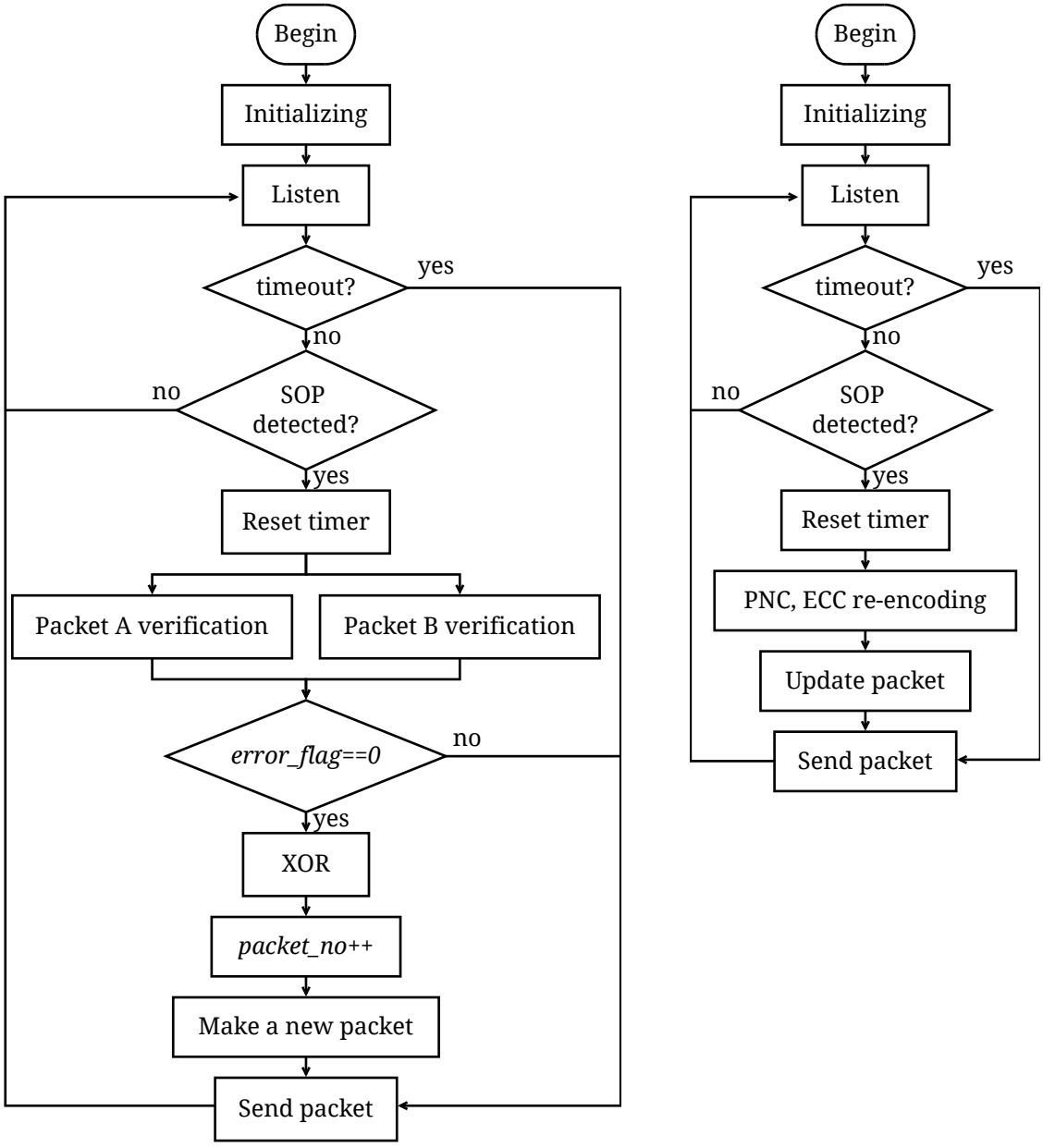
In contrast, the CF relay is unable to demodulate superimposed packets, because it only recovers linear combinations of the transmitted symbols. Therefore, the packet delivers symbols rather than bytes. For this reason, packets transmitted by CF relay have a different format. Namely, they consist of two parts, as illustrated in Figure 4.4(b). The packet begins with a preamble similar to that of conventional packets, followed by the network-coded body, which represents the symbol-wise, frame-synchronous superimposed headers and payloads of both the terminals. The only difference from the conventional preamble is that the last four bytes before the channel estimation training sequence are reserved for the CF linear coefficients a_A and a_B.

4.3 ARQ protocols for different relaying strategies in TWRN

Based on the half-duplex packet switching developed in the previous sections, in this section we describe our design of ARQ protocols for TWRN in GNU Radio with different relaying strategies, namely DNC, DF and CF schemes.

There are two main types of relaying ARQ protocols. First, when the relay participates in the ARQ mechanism, such an ARQ protocol is referred to as relay-terminal ARQ protocol (RT-ARQ). Second, if the relay is not involved in the ARQ mechanism, such an ARQ protocol is referred to as terminal-only ARQ protocol (TO-ARQ). Generally, RT-ARQ protocols outperform TO-ARQ protocols in terms of the network throughput, however the implementation of an RT-ARQ protocol requires complete decoding and additional processing of packets at the relay [111, 112]. The difference between RT-ARQ and TO-ARQ protocols in TWRN relay is illustrated in Figure 4.5.

The performance of different ARQ protocols with PNC was investigated in [113–115]. According to the simulation results provided in [113], RT-ARQ protocols maintained



(a) RT-ARQ (b) TO-ARQ

Figure 4.5: Flowchart of RT-ARQ and TO-ARQ in TWRN. Each relay operates in master mode, therefore it is equipped with a timer which initiates timeout in case no packets were received within T_{out} time interval from the previous transmission.

higher network throughput than other protocols or when no ARQ was utilized, especially when the SNR was medium or high.

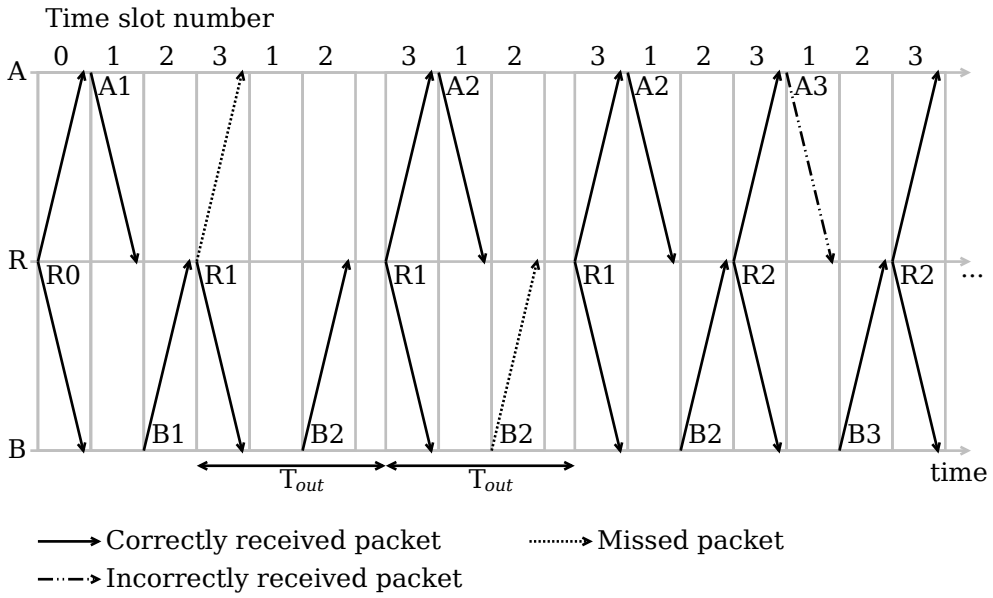


Figure 4.6: Scheme of ARQ protocol for TWRN with DNC relaying. Frame synchronization between terminals A and B is also not required. The relay serves as master, i.e. initializes the TWRN and handles timeouts. Successful reception of two packets results in broadcasting their XORed packet. Otherwise, any error, including packet number mismatch, timeout, missed packet or a packet with uncorrectable errors causes retransmission of the previous XORed packet.

4.3.1 ARQ protocol for DNC

The DNC relay completely recovers each received packet, therefore it allows the implementation of an RT-ARQ protocol. The ARQ protocol for DNC scheme is sketched in Figure 4.6. In order to initialize the protocol, the relay broadcasts a packet containing dummy data (time slot 0). Upon reception, terminal A transmits its packet immediately using time slot 1, while B begins to transmit as soon as the transmission from A is over i.e. time slot 2. If the packets from both the terminals are received successfully, the relay performs bitwise XOR of the data of two packets and broadcasts the resulting packet (time slot 3). After successful reception of this packet, terminals A and B transmit the next packets, and so on. If any packet is lost during any time slot, the timeout at the relay occurs after T_{out} time interval from the previous transmission. In this case, as well as if at least one packet is received with uncorrectable errors or with wrong packet number either from A or B, the relay again broadcasts the previously XORed packet.

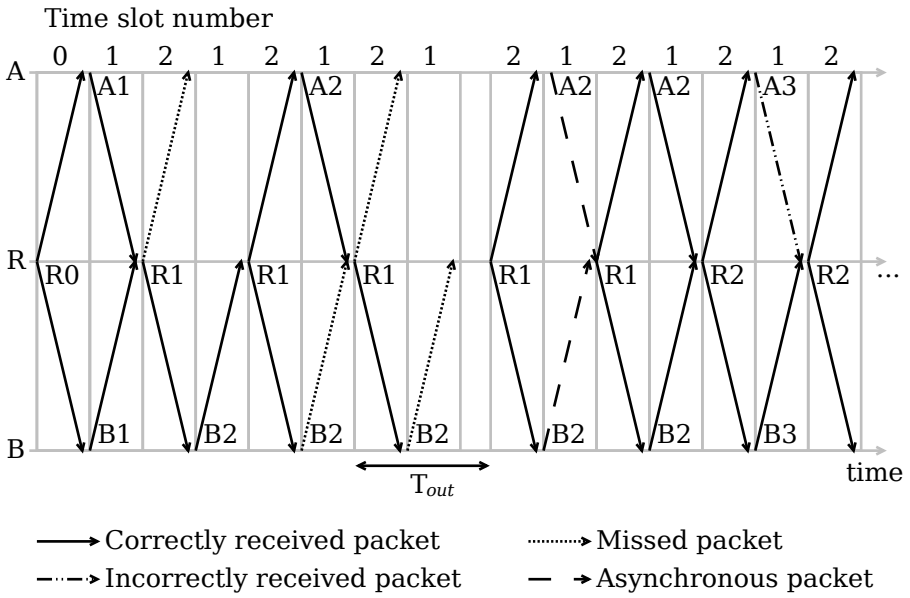


Figure 4.7: Scheme of ARQ protocol for TWRN with DF relaying. Frame synchronization between terminals A and B is required, and failures in frame synchronization cause packets to be missed or decoded incorrectly. The relay serves as master, i.e. initializes the TWRN and handles timeouts. Successful reception of two packets results in broadcasting their XORed packet. Otherwise, any error, including packet number mismatch, timeout, asynchronous packet arrival or a packet with uncorrectable errors causes retransmission of the previous XORed packet.

4.3.2 ARQ protocol for DF

The DF relay is also able to completely recover both packets from one superimposed signal. Hence, it also enables implementation of an RT-ARQ protocol. In fact, with the exception of a few differences related to simultaneous packet arrivals, the DF ARQ protocol is similar to that of the DNC scheme. An example of the ARQ protocol for the DF relaying scheme operation in TWRN is illustrated in Figure 4.7. Again, the communication is initialized by the relay in the same way as in the DNC scheme. Upon reception of the initializing packet, both terminals A and B transmit simultaneously (time slot 1), keeping symbol and frame synchrony. The relay demodulates the received signal and recovers both packets individually, performs error correction, and checks the packets for the presence of uncorrectable errors and insertion/deletion errors, and verifies that the packet ID numbers are as expected. If there are no errors in any packet in the pair, the relay produces the XORed packet and broadcasts it (time slot 2). Upon reception by

the terminals, the two time-slot packet exchange is over. Then the terminals transmit new packets, i.e., the scheme repeats. On the other hand, if there is any kind of error in at least one packet in the received pair, the relay re-broadcasts the previously XORed packet. Thus, "intelligence" of the DF relay supports RT-ARQ protocol functioning and it prevents erroneous packets from being broadcasted. Note that due to the supposed simultaneous packet arrival, the timeout only occurs when there are no replies from both the terminals within the T_{out} time interval. Otherwise, the relay detects the first arrived packet or at least its front part is recovered from a non-superimposed signal and declared an error. However, timeout can also occur when two packets are aligned incorrectly, because failures in the frame synchronization scheme cause incorrect channel estimation and incorrect SOP detection.

4.3.3 ARQ protocol for CF

Unlike DNC and DF relays, which provide bitwise packet recovery and RT-ARQ protocol on top of it, the CF relay only recovers linear combinations of transmitted signals in the physical layer. At the same time, the CF scheme allows implementation of channel decoding and re-encoding of the linear combinations. Therefore, while fully capable on the ECC, the CF relay has limited capability for verifying the packets as DNC and DF relays do. As a result, the CF relay can only support a TO-ARQ protocol. In this case, the corrupted packets also corrupt the linear combination which is forwarded to the terminals, rather than discarded, and the burden of detecting the errors lies with the terminals. This scenario can cause additional retransmissions and decreases the network effective throughput. Figure 4.8 shows an example of the operation of ARQ protocol for the CF relaying scheme in TWRN. The initialization and synchronization are performed in the same way as for the DF scheme, and the same quality synchronization as for the DF scheme is required. The first time slot is also the same, i.e. both packets are transmitted simultaneously. Then, the relay recovers a linear combination of two transmitted signals and re-encodes it with the ECC. As the CF relay is unable to verify the packet number or checksum, it simply forwards the linear combination (time slot 2). The terminals then verify the received packets and transmit a new packet or retransmit the previous packet,

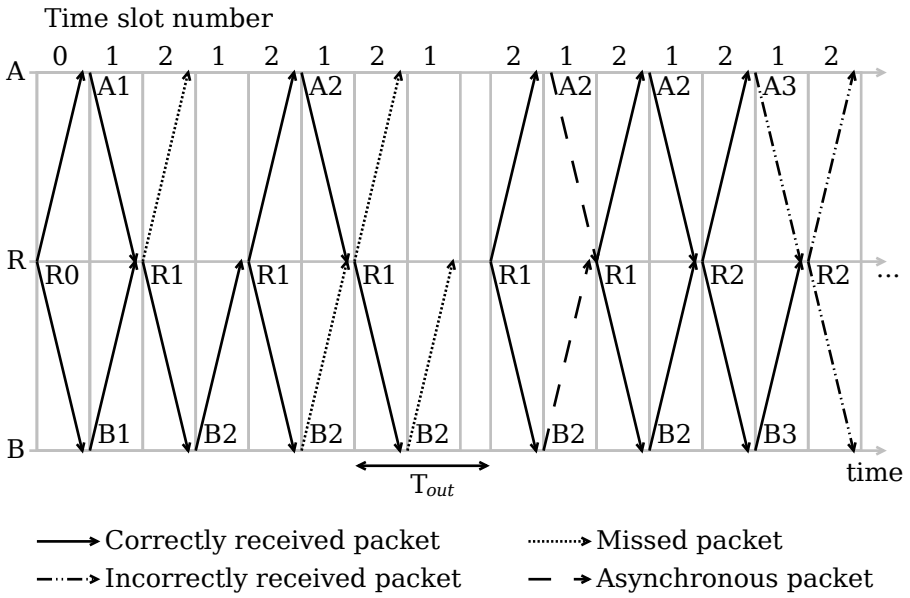


Figure 4.8: Scheme of ARQ protocol for TWRN with CF relaying. Frame synchronization between terminals A and B is required, and failures in frame synchronization cause packets to be missed or decoded incorrectly. The relay serves as master, i.e. initializes the TWRN and handles timeouts. Successful reception of two packets results in broadcasting their linear combination. Errors, including timeout or asynchronous packet arrival, cause retransmission of the previous linear combination. However, errors like packet number mismatch or presence of uncorrectable errors are not detected, and their linear combination is obtained and broadcasted.

i.e. time slot 1 repeats. We note that in the CF scheme the relay also can detect misaligned packets. Therefore, in the same way as in the DF scheme, a timeout only happens when there are no replies from both terminals within the T_{out} interval.

4.4 ARQ performance comparison

In this section we expand simulations presented in Section 3.7, and investigate the performance of different relaying strategies with the proposed ARQ protocols. We consider the DNC, DF and CF relaying strategies, based on the RS$(64, 56)$ code. In this simulations we assume that there are no computational delays between the time slots, but frame asynchrony occurs with probability p_{asynch}. As in the previous simulations, we assume that every receiver is aware of the channel state at any time. The throughputs of TWRN with

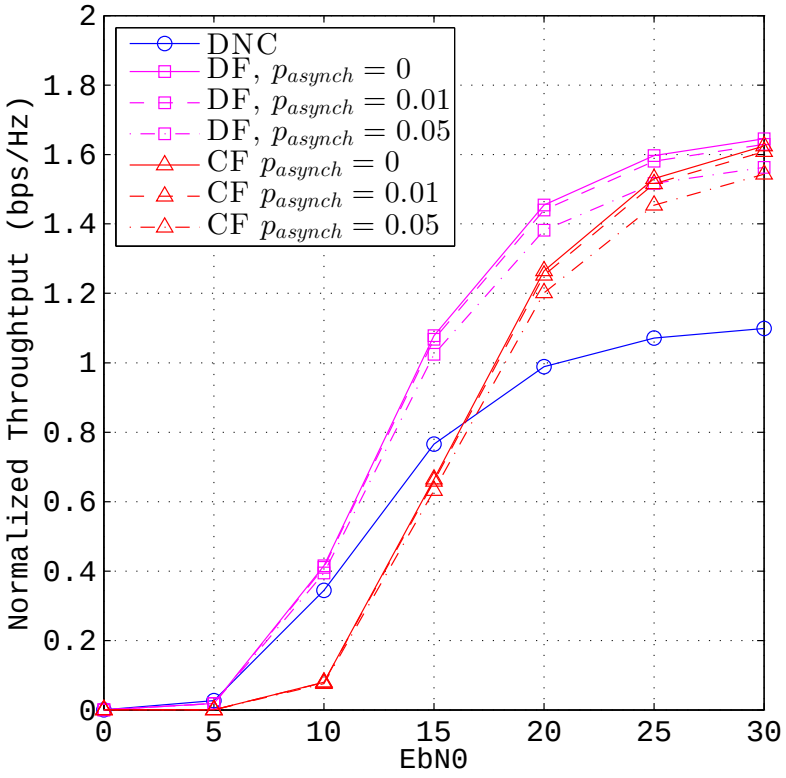


Figure 4.9: Throughput of different relaying strategies in TWRN combined with the relevant ARQ protocol in presence of the frame asynchrony vs. E_b/N_0.

the chosen relaying strategies are compared in Figure 4.9. The solid lines represent the throughput of the respective relaying strategy when the frame asynchrony never happens, i.e. $p_{asynch} = 0$. The dotted lines and dash-dot lines show the throughput of the respective relaying strategy, when the frame asynchrony is 0.01 and 0.05 respectively. Note that since the DNC scheme does not require any synchronization, its performance is independent of p_{asynch}. Our results are consistent with those illustrated in Figure 3.6. We first observe in the figure that the normalized effective throughput of all the schemes is lower than the normalized nominal throughput illustrated in Figure 3.6. This is because of the presence of the overhead, which does not convey the actual data, e.g. the preamble, header and postamble of the packet. Next, it follows from the figure that the presence of frame asynchrony does not cause a significant throughput degradation. As a result, the effective throughput behavior of each strategy matches that of the nominal

throughput. In particular, the DF relaying scheme provides the highest throughput because it takes advantage of performing data exchange in two time slots. The CF scheme also relays the data within two time slots, therefore, with higher energy per bit its performance achieves the performance of the DF scheme. However, with the SNR decrease, the effective noise makes the CF relaying less efficient. Finally, when the E_b/N_0 is 15 dB or less, the throughput of the CF scheme becomes even less than that of the DNC scheme, despite the fact that the DNC relaying scheme takes three time slots per packet exchange.

4.5 Conclusion

In this chapter we have described modifications which we made on the layers above the physical layer in order to incorporate the CF relaying strategy into a real-time communication system. In this chapter we have presented an implementation of logical link control algorithms for the relaying strategies employed in the testbed such as the DNC, DF, and CF schemes. The developed ARQ protocols are designed to take into account GNU Radio natural random delay and other imperfections. The performance of these ARQ protocols is evaluated experimentally in the next chapter.

Chapter 5

SDR implementation of CF relaying and experimental evaluation

In this chapter we present the first real-time GNU Radio implementation of a two-way relay network with compute-and-forward (CF) relaying strategy. Despite the fact that the theoretical results for CF are promising, the implementation is hindered by a number of practical problems. We first identify these problems and then propose our solutions to minimize their impact. Rather than developing complex algorithms able to cope with certain types of asynchrony, we propose a synchronization scheme for GNU Radio, a popular software-defined radio (SDR) platform. The scheme enables simple implementation of synchronous CF algorithms. With the implemented prototype working in real-time we are able to conduct experimental performance analysis of the CF scheme. Our experimental results show that when the SNR is high, CF relaying outperforms other relaying strategies in terms of the network throughput. Therefore, our testbed experimentally verifies the benefits of CF relaying predicted by the theoretical analysis. With the common practical problems solved, the testbed implemented in GNU Radio allows rapid modification for physical layer network coding (PNC) algorithms other than CF, thus simplifying their experimental analysis.

5.1 Introduction

IN theory, when the SNR is high, PNC can increase the TWRN throughput by 50% compared to that of digital network coding (DNC) [7]. In contrast, DNC is expected to outperform PNC in low SNR regimes. In addition to low SNR, several other factors negatively affect the practical performance of PNC. First, processing delays are caused by

the fact that PNC is more computationally intensive than DNC. Second, synchronization delays are introduced by the need to synchronize the packets transmitted at the different terminals. Finally, a PNC relay does not recover the packet data, and therefore does not provide any packet verification capabilities, unlike a DNC relay. This causes the relaying of corrupted packets and requires more retransmissions. All these factors reduce the actual performance compared to the theoretical performance of the TWRN. However, their impact cannot be evaluated adequately in simulations, because there are no accurate models for these factors. In contrast, the impact of the above factors can be experimentally estimated if a relay network testbed with PNC is implemented.

In this chapter, we summarize our design of a TWRN testbed in GNU Radio, where the relaying is performed with three different strategies, namely the DNC strategy, the MIMO-based decode-and-forward (DF) and the modified CF scheme developed in Chapter 3. To the best of our knowledge, our testbed is the first real-time implementation of the CF relaying strategy. Our main contribution is that, unlike the previous work, we propose practical symbol and frame synchronization schemes which provide sufficient synchronization for the standard PNC algorithms designed under the synchrony assumption, and implement the schemes on FPGA of USRP N210. The synchronization scheme thus allows us to avoid using the computationally intensive asynchronous algorithms reviewed in Section 2.4. We also describe implementation challenges specific to the GNU Radio platform and our solutions applicable for all the strategies, such as channel estimation methods and symbol-timing recovery for superimposed PNC signals. In order to make the testbed work in real time, we implement the half-duplex packet switching and ARQ protocols described in Chapter 4. Utilizing these schemes and solutions, one can easily repeat our implementation on the GNU Radio platform or customize it according to one's needs at no additional cost. Furthermore, our implementation on the GNU Radio platform can be easily customized and extended to higher-order modulation schemes such as M-QAM or more complex networks beyond TWRN.

In addition, we evaluate the performance of different relaying strategies with several experiments. We demonstrate that with high SNR, the practical throughput of the CF scheme is higher than that of the other relaying strategies, despite some practical imple-

mentation factors degrading performance. In the presence of these factors, the CF scheme achieves practical throughput improvement of about 30% rather than the 50% predicted by theoretical analysis. On the other hand, the experiments show that the DNC relaying scheme is more reliable at low SNR.

The rest of this chapter is organized as follows. We begin this chapter with the detailed description of the SDR platform employed in the testbed provided in Section 5.2. Next, in Section 5.3 we summarize the synchronization requirements and introduce our symbol and frame synchronization scheme, as well as the implementation of these schemes on the USRP. In Sections 5.4 and 5.5 respectively, we present our approaches to symbol-timing recovery and channel estimation for PNC/CF relays. The experimental results are provided, and the advantages and drawbacks of different relaying strategies are discussed in Section 5.6. Finally, Section 5.7 concludes this chapter.

5.2 Hardware and software platform

Each terminal and relay is realized with a USRP N210 by Ettus Research connected to an individual PC running GNU Radio. The details of the main components of the testbed are provided in Table 5.1. The USRPs are equipped with XCVR2450 half-duplex daughterboards operating in $2.4 - 2.5$ GHz and $4.9 - 5.9$ GHz dual band. Each USRP is connected to the host PC with the GigE interface, which provides up to 25 MSPS data throughput. Each USRP is also equipped with a GPS disciplined oscillator module (GPSDO), by the same manufacturer. In order to implement a MIMO receiver for the DF scheme, the DF

Table 5.1: Details of the Main Components of the Testbed.

Component	Details
CPU	Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
NIC	NetXtreme BCM5722 Gigabit Ethernet PCI Express
OS	Ubuntu 14.04 LTS, 64 bit
GNU Radio	v.3.7.4
UHD	v.003.007.001
SDR Hardware	Ettus USRP N210

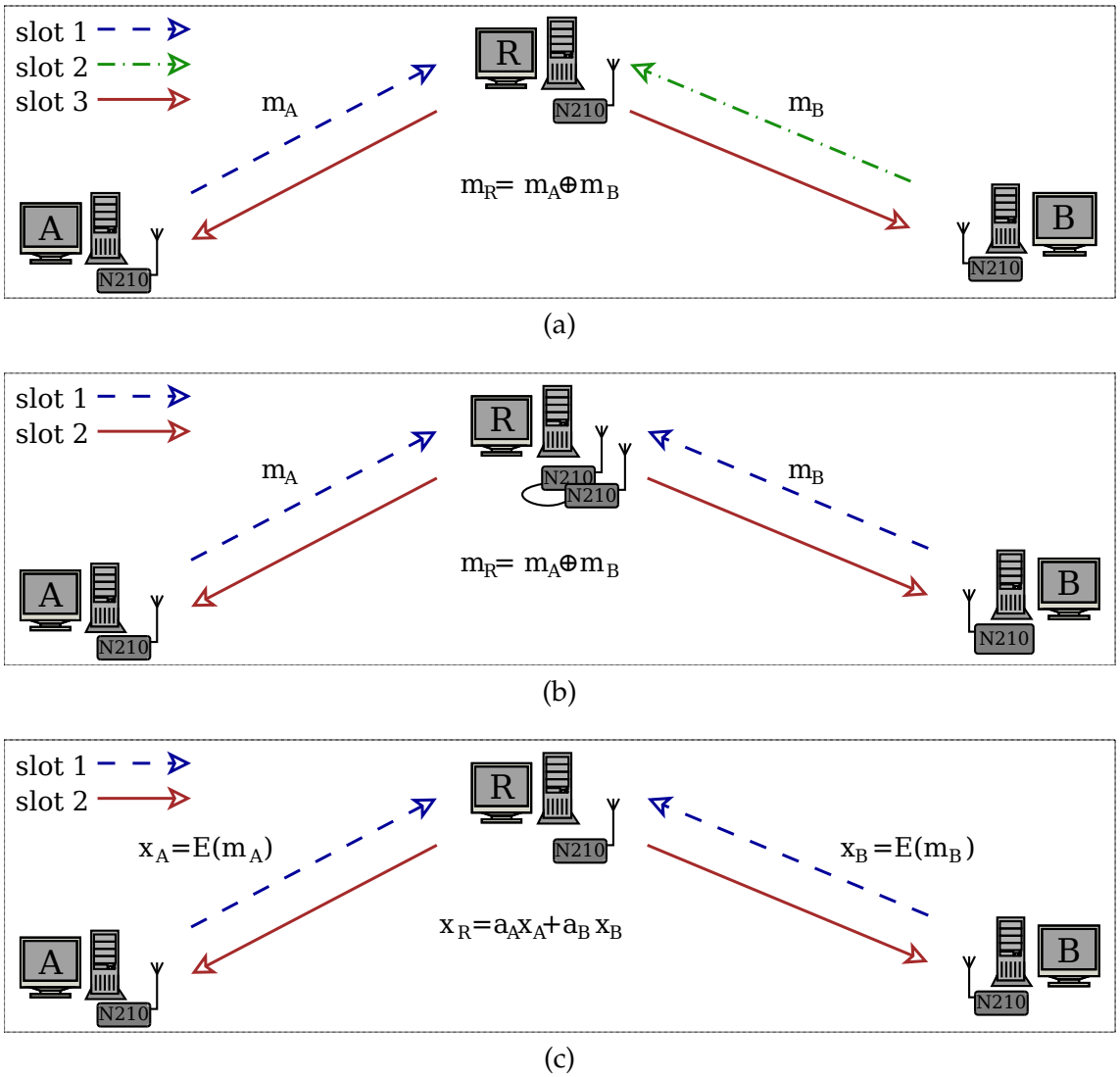


Figure 5.1: Architecture of our TWRN testbed with different relaying strategies: (a) DNC, (b) DF, and (c) CF. The DF relay is MIMO-capable, i.e. two USRP N210 devices connected with MIMO cable, while DNC and CF relays as well as all terminals are single antenna devices, implemented on single USRP.

relay is equipped with two USRPs connected through a MIMO cable. Figure 5.1 illustrates the three testbed architectures for each of the relaying strategies implemented in the testbed.

5.3 Synchronization

Symbol and frame synchronization is a crucial requirement for the implementation of PNC algorithms. In the literature review, in Section 2.4, we analyzed several asynchronous PNC and CF algorithms from the point of view of their implementation simplicity in SDR. The algorithms proposed in the literature require significant modification of the original algorithms designed based on the synchrony assumption. The modifications, in turn, make those modified algorithms not universal, i.e. robust against one particular asynchrony, thus limiting the scenarios where the algorithms can be utilized. Furthermore, the asynchronous algorithms are usually more computationally complex. These drawbacks motivate us to choose another approach in this work: we first implement synchronization mechanisms which provide sufficient synchronization for the implementation of synchronous CF algorithms, such as in [48,49] and our CF scheme described in Chapter 3. In this section we present our GPS-assisted symbol and frame synchronization schemes and then discuss FPGA customization required for the implementation of these schemes.

5.3.1 Symbol synchronization

In this testbed we implement a GPS-assisted symbol synchronization scheme. First, we equipped each terminal with the GPSDO modules [116]. A GPSDO, when locked to the GPS satellites, is able to retrieve the absolute GPS time. We then configured the USRPs to use the GPSDO as a clock and time source. As a result, the internal USRP time is initialized to the GPS time. According to the specifications, the root-mean-square error of such synchronization is less than 50 ns, i.e. much smaller than the typical symbol duration supported by GNU Radio (640 ns in our case). Therefore, the GPS-assisted symbol synchronization accuracy is sufficient for the implementation of PNC algorithms.

The configuration of GPSDOs to be used as a clock and time source is accomplished by properly setting up the USRP hardware according to [117], followed by configurations of the USRP Sink and USRP Source blocks in GRC. Note that the symbol synchronization is only required between the terminals, and is not necessary between a terminal and the relay. Although also equipped with a GPSDO module, the relay only utilizes it for cor-

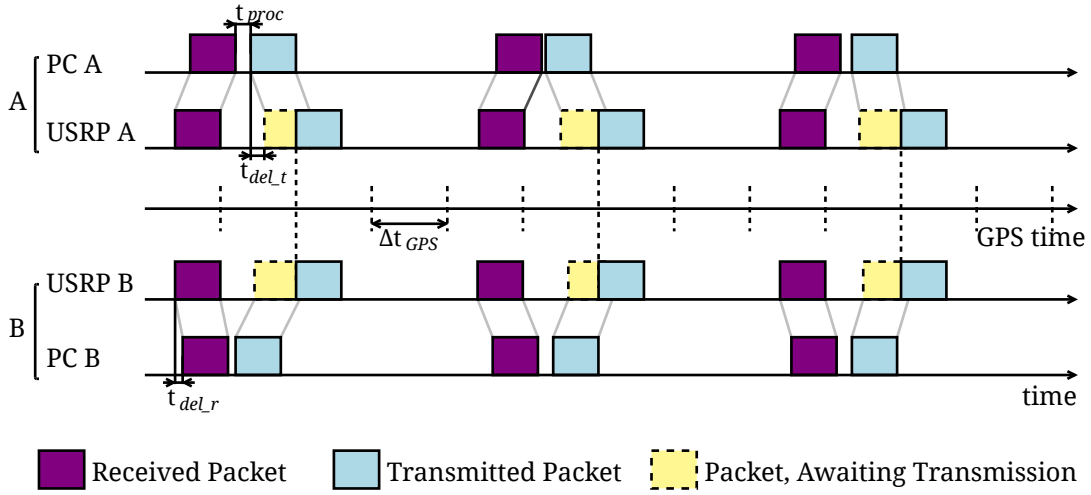


Figure 5.2: Frame synchronization scheme for DF and CF TWRN. t_{del_r} represents the time interval needed to transfer a received packet from the USRP to the host PC. t_{del_t} represents the time interval needed to transfer a transmitted packet from the host PC to the USRP. t_{proc} is the time interval required for a received packet verification and generating a response packet to be transmitted. The duration of t_{del_r}, t_{del_t} and t_{proc} is random due to the scheduler's random queuing delay.

rection of the local oscillator (LO) frequency offset.

Note that a USRP with XCVR2450 daughterboard is mainly appropriate for short-range wireless communication, where the propagation delay is negligible. Therefore, we assume that the packets transmitted simultaneously arrive at the same time. In general, however, one terminal could be located much further from the relay than the other terminal, and the large difference between the distances may cause a significant offset between two packets transmitted simultaneously. This offset can be compensated if the terminal nearest to the relay transmits after a short delay equal to a few clock periods of its FPGA. For example, if the FPGA clock period is 100 ns, as in USRP N210, a distance difference of 60 meters can be compensated by introducing a two clock-period delay in the terminal nearest to the relay.

5.3.2 Frame synchronization

Frame synchronism, i.e., simultaneous arrival of packets from both terminals, is another necessary condition for PNC implementation. Frame synchronization in GNU Radio is

hindered by its inherently random processing time. Additional time is needed to transfer packets from the USRP to the PC and back via the Ethernet cable. These delays cause asynchronous processing of the received packets and asynchronous delivery of the packets to be transmitted to the USRP.

Moreover, *underflow* exceptions also contribute to the loss of frame synchronism. The host PC supplies the data to USRP in blocks of the length defined by the GNU Radio scheduler. In such a way, one packet may be sent to the USRP by several blocks. The underflow happens when the host PC is not able to provide the next block fast enough immediately after the previous block has been transmitted. When the FPGA detects an underflow, it waits until the next block arrives and then retransmits the previous block followed by the next block. Therefore, underflows are undesirable as they distort the frame synchronization in the middle of transmission.

Once the terminals are symbol-level synchronized, as described in the previous subsection, we focus on the frame-level synchronization. Below we summarize our method to achieve frame synchronization, given the aforementioned problems:

1. We have relocated a few signal processing routines from the GNU Radio on the host PC to the USRP's FPGA. Those routines include modulation and pulse-shaping filtering in the FPGA's Tx chain, and automatic gain control (AGC) and matched filtering in the Rx chain. The reader is referred to Section 5.3.3 for details of FPGA customization. Assuming QPSK modulation, the relocation reduces the number of bytes per packet sent via the Ethernet cable by a factor of 32. As a result, the number of underflows in our experiments was reduced by more than 10 times. Both the matched filter and the pulse-shaping filter are represented by the root-raised cosine (RRC) filter. Due to the limited number of multipliers in the FPGA of USRP N210, the design and implementation of the RRC filter is to be without multipliers. We review existing methods of multiplierless filter design and propose a new method based on zero/pole approximation of an IIR filter, also suitable for the RRC filter multiplierless implementation on FPGA, in Chapter 6.

2. Some blocks cannot be relocated, for instance those which require floating-point multiplications, such as channel estimation modules. In order to support higher

sampling rates, we make use of Vector-Optimized Library of Kernels (VOLK) [100], wherever possible. This library simplifies the use Single Instruction Multiple Data (SIMD) instructions. A SIMD instruction performs operations on vectors rather than on scalars, thus accelerating computations significantly. For example, element-wise multiplication of floating-point complex vectors with VOLK can be performed more than five times faster, compared to the use of non-vectorized multiplication. In addition, the matrix computations involved in finding the shortest vector coefficients with CLLL are implemented with the Armadillo C++ high-speed numeric library [118].

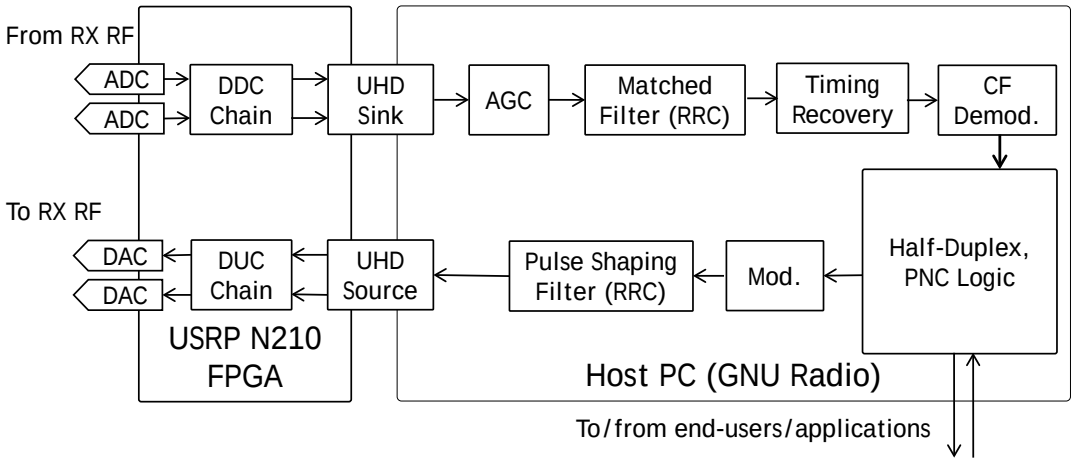
3. We have implemented a new frame synchronization scheme, where the GPS-assisted synchronization is carried out by the USRPs of the terminals. This scheme is presented in Figure 5.2. Despite the fact a packet transmitted by the relay arrives at both terminals' USRPs simultaneously, each host PC receives the packet with random delay t_{del_r}. Consequently, the response packet to be transmitted is generated after delay t_{proc} and arrives at the USRP after another random delay t_{del_t}. Therefore, although the terminals received the packet simultaneously, they would not start the reply transmission synchronously. However, both USRPs are to start transmitting simultaneously, in order to ensure successful PNC/CF operation. For achieving this requirement, the synchronization scheme only allows the USRPs to transmit after a fixed delay Δt_{GPS} from the previous transmission. Note that the USRPs are already symbol-synchronous and their internal time is also synchronized with the GPS time signal. We have modified the FPGA image in such a way, that upon reception of a packet from the host PC, the USRP holds it until the Δt_{GPS} time interval expires. Thus, the frame synchronization is achieved. The choice of Δt_{GPS} is determined taking into account the total delay $t_{del} = t_{del\_r} + t_{del\_t} + t_{proc}$ by the following tradeoff. If Δt_{GPS} is too large compared to the average t_{del}, the chance that two terminals can receive the packet and reply within Δt_{GPS} is high, therefore the chance of a non-synchronized transmission is low. However, large Δt_{GPS} causes the throughput decrease. On the other hand, when the Δt_{GPS} is short, the chance of a non-synchronized transmission increases, making decoding of incorrectly super-

imposed packets impossible.

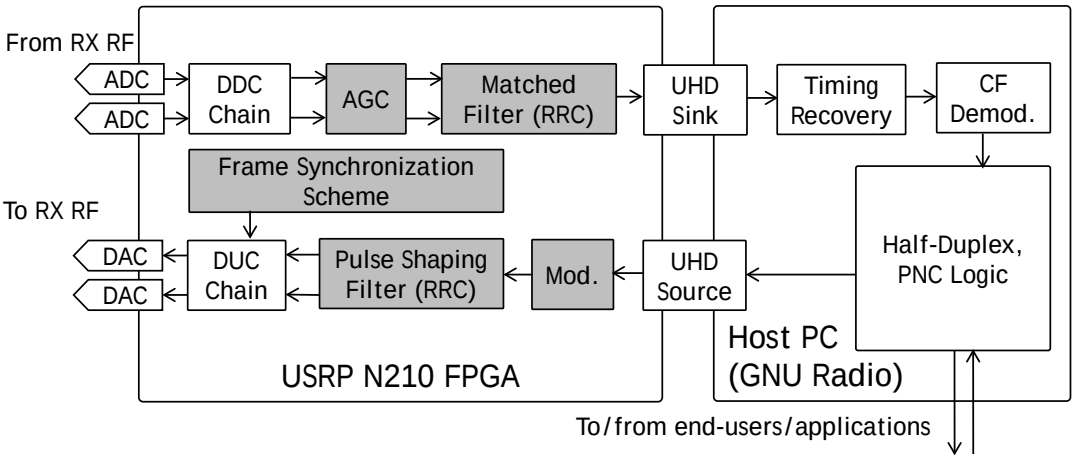
5.3.3 FPGA customization

In order to be able to cope with all sorts of delay introduced by both USRP and GNU Radio Tx and Rx, the frame synchronization scheme described in the previous subsection should be implemented as close as possible to the RF front-end, i.e. on FPGA. Therefore, implementation of the frame synchronization scheme requires modification of the standard FPGA image of USRP N210 provided by Ettus Research. To achieve this, we modified the FPGA Verilog source code provided along with the UHD software, and synthesized a new FPGA images for the testbed. In addition, we relocated a few signal processing routines from GNU Radio to FPGA in order to reduce the computational load on GNU Radio and avoid underflows. These modifications were implemented and new FPGA images was synthesized with Xilinx ISE Design suite 14.5.

Figure 5.3 compares the original architecture of the SDR system i.e. USRP FPGA, UHD and GNU Radio with the modifications we implemented in this testbed. For simplicity, we omit illustration of components such as FIFOs, buffers, Ethernet controllers and RF front-end which we did not modify. In the original architecture shown in Figure 5.3(a) the FPGA only performs DDC in the Rx chain and DUC in the Tx chain, the tasks common and compulsory for all applications. In other words, the default architecture of USRP FPGA leaves any application-specific signal processing algorithms to be implemented on the host PC in GNU Radio. As mentioned previously, relocation of a few signal processing routines from the PC to the FPGA reduces processing delays and randomness of the delays, which result in underflows. In this way, implementation of the pulse-shaping filter in the Tx chain and the matched filter in the Rx chain, both as RRC filter seems computationally intensive on the host PC, yet is feasible on the FPGA. In addition, the AGC block in the Rx chain and the modulator in the Tx chain are also good candidates for relocation to the FPGA, because they do not require intensive floating-point multiplications. Finally, the frame synchronization scheme, described in the previous subsection, is also located on the FPGA. The updated architecture of the FPGA image is sketched in Figure 5.3(b).



(a)



(b)

Figure 5.3: FPGA Customization for Frame Synchronization. (a) Original architecture as provided by Ettus Research and GNU Radio, (b) Our improvements: the AGC and matched filter of Rx as well as the pulse-shaping filter, modulator and frame synchronization scheme of Tx are relocated from the host PC to FPGA. The relocated blocks are highlighted in grey.

One particular advantage of the presented implementation is that the entire Tx chain is moved from the host PC to the FPGA. As a result, a packet to be transmitted can be transferred to the UHD and subsequently to the FPGA as quickly as possible, most likely within one block (from the scheduler point of view). Therefore, the impact of the GNU Radio delays is minimizing, which in turn minimizes the chance of any underflow occur-

rence.

The performance of the hardware and the latency could be further reduced if the symbol-timing recovery module is also implemented on the FPGA. However, the Gardner algorithm requires performing intensive floating-point multiplications for timing-error detection and fractional resampling (see Section 5.4), which cannot be implemented on the FPGA of USRP N210 due to the limited number of only 18-bit fixed-point multipliers. This means that the Gardner algorithm can only be implemented on FPGA of any newer generation Ettus SDR products such as USRP X series which are equipped with more powerful FPGA.

5.4 Symbol-timing recovery at CF relay

The symbol-timing asynchrony appears from the fact that a receiver is unaware of the precise arrival time of the pulses. Due to this fact, nonzero timing delay τ_d between the optimal sampling time and the receiver's ADC clock, which is unknown by the receiver, usually exists. In order to take into account τ_d, the received signal $r(t)$ is modeled as follows

$$r\left(t\right) = hx\left(t - \tau_d\right) + v(t), \tag{5.1}$$

where h is the channel coefficient, $x(t)$ is the transmitted signal with symbol period T and $v(t)$ is AWGN. We assume that $0 \leqslant \tau_d \leqslant T$. A symbol-timing recovery algorithm attempts to estimate and track the timing delay τ_d. Sampling at time points $t_s = kT + \hat{\tau}_d$, where $k \in \mathbb{Z}^+$, maximizes SNR at the receiver and minimizes intersymbol interference (ISI), hence the timing delay should be estimated accurately.

In modern software receivers sampling the received signal $r\left(t\right)$ at the optimal time $t_s = kT + \hat{\tau}_d$ is inefficient from the perspective of hardware because of the need to adjust t_s with changing $\hat{\tau}_d$. Instead, $r\left(t\right)$ is sampled with a fixed sampling period $T_s < T/2$, i.e. with oversampling. After proper amplification and matched filtering, the output of matched filter s_n is fed into fractional interpolator [120] which produces estimates of timing delay $\hat{\tau}_d$ and received signal at the optimal sampling time $\hat{s}\left(kT + \hat{\tau}_d\right)$ fully in a digital manner. Subsequently, output of the fractional interpolator is passed to the demodulator

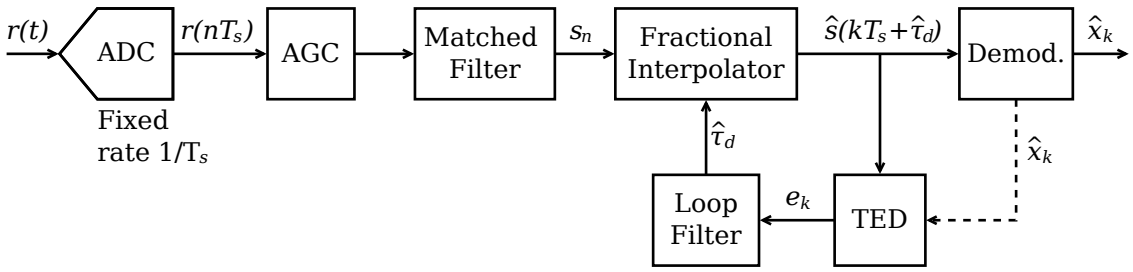


Figure 5.4: Feedback symbol-timing recovery scheme [119]. The received signal $r(t)$ is sampled with fixed sampling period $T_s < T/2$. The amplified and filtered signal $s_n = s(nT_s)$, $n \in \mathbb{Z}^+$ is fed into the fractional interpolator which estimates τ_d and samples $\hat{s}\left(kT + \hat{\tau}_d\right)$, $k \in \mathbb{Z}^+$. The samples are used for the demodulation and by the TED. The TED estimates timing error $e[k]$ needed for update of $\hat{\tau}_d$. The dashed lines represent the optional components when TED is decision-directed, such as MM algorithm.

or equalizer. Figure 5.4 illustrates such a fully digital symbol-timing recovery scheme based on feedback timing error detection (TED), where the output of the fractional interpolator is also used for estimating the timing error e_k, needed for updating $\hat{\tau}_d$. Other schemes, such as feed-forward TED are discussed in [119].

Of all methods of feedback symbol-timing recovery, the modified Mueller and Muller algorithm (MM) [121] and Gardner algorithm [122] are two popular algorithms used in software receivers. The difference between them is how they perform TED. In the MM algorithm the timing error is estimated as follows

$$\begin{aligned} e_k^{MM} &= \left(\hat{s}_I\big(\,(k-1)\,T + \hat{\tau}_d\big) - \hat{s}_I\left(kT + \hat{\tau}_d\right)\right)\Re\left(\hat{x}_k\right) \\ &+ \left(\hat{s}_Q\big(\,(k-1)\,T + \hat{\tau}_d\big) - \hat{s}_Q\left(kT + \hat{\tau}_d\right)\right)\Im\left(\hat{x}_k\right), \end{aligned} \tag{5.2}$$

where $\hat{s}_I(t)$ and $\hat{s}_Q(t)$ are I and Q components of $\hat{s}(t)$, $\hat{x}_k$ is the demodulator output i.e. decision based on sampled value $\hat{s}\left(kT + \hat{\tau}_d\right)$. In the Gardner algorithm, the timing error is

$$\begin{aligned} e_k^{G} &= \left(\hat{s}_I\big(\,(k-1)\,T + \hat{\tau}_d\big) - \hat{s}_I\left(kT + \hat{\tau}_d\right)\right)\hat{s}_I\left(kT - T/2 + \hat{\tau}_d\right) \\ &+ \left(\hat{s}_Q\big(\,(k-1)\,T + \hat{\tau}_d\big) - \hat{s}_Q\left(kT + \hat{\tau}_d\right)\right)\hat{s}_Q\left(kT - T/2 + \hat{\tau}_d\right). \end{aligned} \tag{5.3}$$

A comparison of (5.2) and (5.3) shows the advantages and disadvantages of each method. The MM algorithm is decision-directed, hence it is sensitive to the carrier offset and phase errors. Therefore, the carrier and phase recovery should be performed before the timing recovery. However, once the carrier and phase offsets are recovered, the MM algorithm provides faster timing recovery, and is less sensitive to low SNR than non-decision-directed methods. Furthermore, the MM algorithm can be enhanced, if the timing recovery is performed in cooperation with channel decoding [123]. At the same time, the MM algorithm demonstrates reasonable performance with PSK modulations only. In contrast, the Gardner algorithm is non-decision-directed, and is therefore robust against the carrier frequency and phase offsets. Therefore, the channel estimation can be performed after the timing recovery. The Gardner algorithm can also be applied for receivers where the constellation of a received signal is larger than a PSK constellation. However, the Gardner method suffers from self-noise, especially when the SNR of the received signal is low. The impact of self-noise can be reduced with interpolation with a higher upsampling factor [124, 125]. The Gardner method also requires estimation of $s(t)$ with half symbol period delay after the optimal sampling time i.e. $\hat{s}\,(kT - T/2 + \hat{\tau}_d)$, thus the fractional interpolator should produce twice more output. In addition, the output of the Gardner TED is proportional to the square of magnitude of the input signal. For this reason, accurate AGC should precede the Gardner symbol-timing recovery block.

The problem of symbol-timing recovery for conventional communications has been studied in detail. However, to the best of our knowledge, this problem in context of PNC, i.e. symbol-timing recovery for superimposed signals, has still been little studied, with only a few results based on realistic assumptions being available [126, 127]. For example, although [126] offers a symbol-timing recovery solution for non-OFDM communications, the proposed method requires high oversampling and the use of computationally complex DFT-based interpolation. This problem is also not covered in studies describing implemented PNC prototypes [32, 74, 75].

In our testbed we utilize another approach. Assuming that with the proposed GPSDO-

based synchronization the terminals are symbol-synchronous, i.e.

$$\tau_d = \tau_d^A = \tau_d^B, \tag{5.4}$$

the relay only estimates one parameter τ_d at the beginning of each packet. Therefore, we can perform the PNC symbol-timing recovery with a method used for conventional symbol-timing recovery. Given that the superimposed constellations at the CF receiver are larger than PSK constellations and considering other advantages of the Gardner algorithm discussed above, we adopt the Gardner symbol-timing recovery method in our testbed.

As data exchange in TWRN is packet-based rather than continuous, the timing recovery algorithm is expected to provide both fast acquisition of $\hat{\tau}_d$ at the beginning of each packet and quick updates of timing delay $\hat{\tau}_d$ if a change occurs in the middle of a packet. For the acquisition phase we add a symbol-timing recovery preamble (STRP) in front of each packet. We then experimentally determine the required length of the STRP. The three plots in Figure 5.5 demonstrate an example of symbol-timing recovery in the CF relay. In this example $T_s = T/2$, i.e. the oversampling is 2 samples/symbol. Before the packet arrives, the estimate of timing delay is set to one-half symbol period T, $\hat{\tau}_d/T = 0.5$. During the acquisition phase the TED estimates that the timing error e_k^G is positive. Accordingly, $\hat{\tau}_d$ is updated in a way to minimize the timing error. With the estimate $\hat{\tau}_d$ becoming closer to the actual delay τ_d the timing error is minimized and centered around zero.

In the example demonstrated in Figure 5.5, the acquisition phase takes about 200 symbols, however, the SNR is high. Based on several experiments performed in different SNR regimes we determined that a 150-200 bytes long STRP (600-800 symbols) in front of a packet is sufficient to provide symbol-timing recovery at the CF relay in lower SNR regimes, without affecting the main part of the packet.

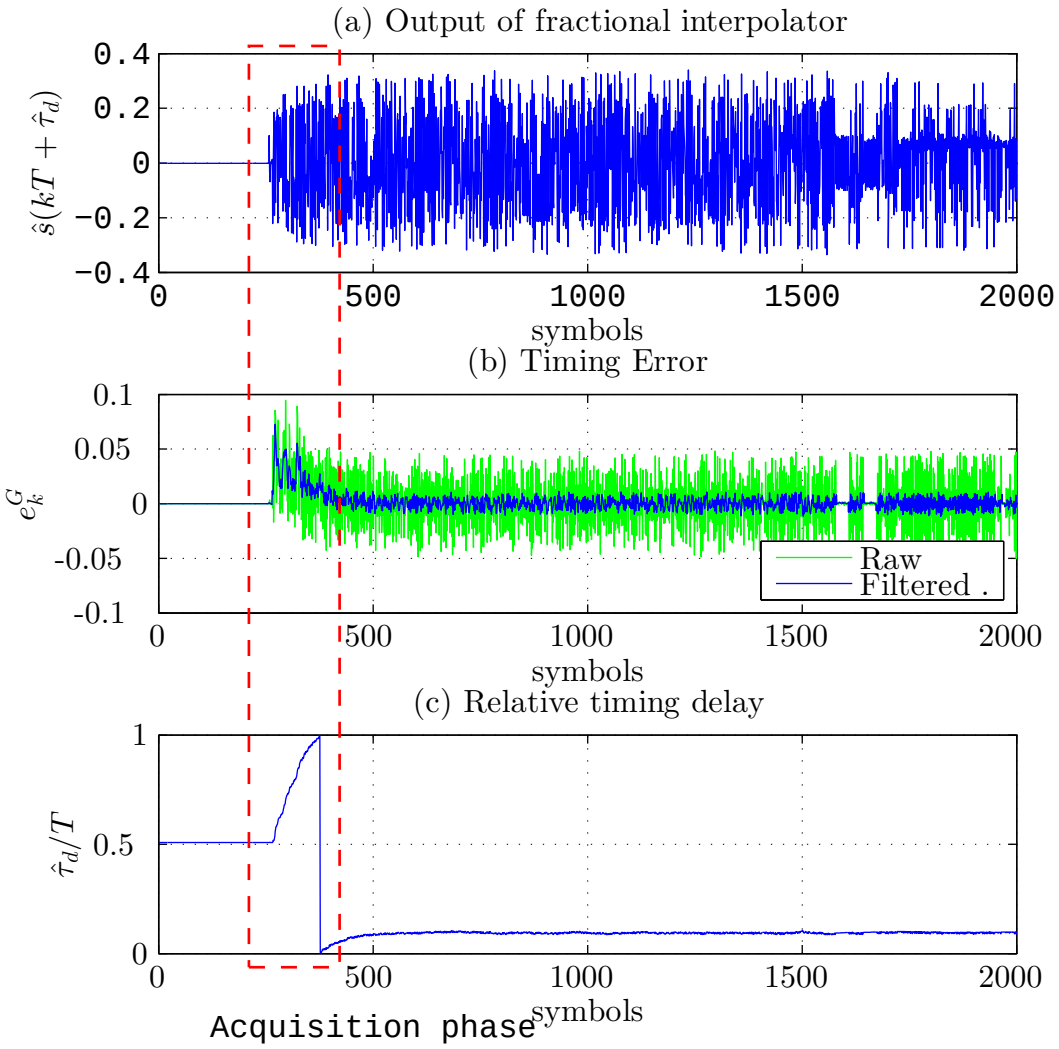


Figure 5.5: Example of timing recovery in CF relay.

5.5 Channel estimation

All the relaying strategies implemented in this testbed assume that the relay knows the channel coefficients h_{AR} and h_{BR}. Hence, the relay should be able to obtain accurate estimates of the channel coefficients from the arriving packets. The terminals should also estimate their channel coefficients based on the received packets.

In our testbed, a training sequence which consists of 32 symbols, known by the desired receiver, is placed in the preamble of each packet after the STRP preamble. This se-

quence is used for channel estimation with conventional least-squares estimation methods in all terminals and the DNC relay. The DF and CF relays, however, estimate the channel coefficients from the superimposed packets. Therefore, in the DF and CF relays we have applied techniques from MIMO channel estimation (2.15) and (2.16), assuming the packets are synchronized with the proposed synchronization scheme. Note that both training sequences $\mathbf{t}_A$ and $\mathbf{t}_B$ are also modulated with the QPSK modulation scheme. The training sequences are generated randomly in the beginning of experiments, however they are to be orthogonal and of the same norm, as recommended in [68]. Consequently, we have implemented the joint channel estimation algorithm as a part of the related GNU Radio blocks.

Our experimental results demonstrate that the implemented method of channel estimation provides accurate estimates of channel coefficients h_{AR} and h_{BR} when the symbol-timing is estimated correctly before the channel estimation.

The carrier frequency asynchrony is also effectively mitigated with the use of GPSDO. The GPSDO module provides the reference signal to the local oscillator, thus constantly correcting any drift away from the target frequency. The measurement results showed that the residual frequency offset was only about 4 Hz. Therefore, in this work we can assume that the channel rotation is slow, i.e. almost negligible within the duration of one packet. Therefore, once estimated at the beginning of packet, the estimates of channel coefficients $\hat{h}_{AR}$ and $\hat{h}_{BR}$, as well as CF integer coefficients a_A and a_B do not need to be updated in the middle of the packet.

5.6 Experimental performance evaluation

In this section, we demonstrate the performance of the testbed with the CF, DF and DNC schemes via a series of experiments in the indoor environment. Note that the detail CF scheme design is discussed in Chapters 3 and 4, the DF scheme design provided in Section 2.2.1 and Chapter 4, and the DNC scheme is given by (1.1) and (1.2) and its data-link layer protocol is described in Chapter 4. The data exchanged between the terminals were randomly generated.

Table 5.2: Summary of Main Parameters for GNU Radio Blocks.

Block	Parameter	Value
USRP Source, USRP Sink	Sampling Rate Center Frequency LO Offset Antenna Mb0: Clock Source Mb0: Time Source	3.125 MSPS 2.41 GHz 2 MHz J1 O/B GPSDO O/B GPSDO
USRP Source	Gain	0 dB
Mod./Demod.	Modulation Tx Digital Gain [b], g_d PS Filter Oversampling Roll-off of the RRC AGC	QPSK (Gray code) 0.2 RRC Filter 2 samples /symbol 0.3 Default
Half-duplex	Packet Length Timeout T_{out} ECC	4096 bytes 0.1 s. (64,56) Reed-Solomon, Adler-32 checksum

[a] The input level of signal for the USRP Sink must be within the range of $[-1, 1]$ to avoid arithmetic overflow when converting from *float* to *short* type. To guarantee this, the input is multiplied by a coefficient $0 < g_d < 1$. In this work g_d is referred to as *Tx Digital Gain*.

5.6.1 Experimental set-up

Table 5.2 contains the set-up of the main parameters for various GNU Radio blocks involved in the communication. In particular, we set the GNU Radio sampling frequency to 3.125 MSPS. This means that the USRP's ADC and DAC sampling rate was still 100 MSPS, however the Rx signal was downsampled by a factor of 32 in the DDC chain and the Tx signal was upsampled by the same factor in the DUC chain. This GNU Radio sampling frequency with oversampling 2 sample/symbol resulted in a symbol rate of 1.5625 MBd. Even though the testbed supports higher sampling rates up to 12.5 MSPS, the chosen sampling rate provides smooth operation of the testbed, thus CF performance analysis is less affected by the GNU Radio delays and residual symbol asynchrony[1] .

[1]Shortly after submission of this thesis, we have developed a new enchanced symbol-level synchronization scheme for USRP, which increases the stability of the testbed at higher sampling rates [128].

Our experiments were conducted in the laboratory environment, such that the distance between each terminal and the relay was approximately 4 meters. In order to obtain experimental results with both high and low SNR we set the Rx gain to 0 dB and limited the Tx gain to 22 dB. We also reduced the Tx digital gain g_d to 0.2.

In these experiments our goal is the performance evaluation of the relaying strategies against the Tx power. The problem with all USRP devices is that since the daughterboards are not well calibrated, the absolute Tx power may differ slightly from one device to another. Furthermore, the absolute Tx power may depend on many factors, such as the frequency, ambient temperature and stability of the power supply. However, the user can control the approximate Tx power by setting the Tx gain (i.e. the gain of the Tx amplifier) and the Tx digital gain g_d as explained in Table 5.2. Therefore, we fix the Tx digital gain and plot the Tx gain on the X-axis measured in dB, rather than the absolute Tx power measured in dBm. In our experiment, when measured with a spectrum analyzer, the Tx gain of 22 dB approximately matches the absolute Tx power of 4 dBm, while the Tx gain of 4 dB matches the Tx power of about -10 dBm. If the measurement is repeated with other XCVR2450 daughterboards, the level of Tx power will be slightly different but should not deviate much.

In addition, XCVR2450 transmitters expose a significant DC component, that varies from one daughterboard to another and causes a decrease of the effective Tx SNR [73]. In order to improve the SNR we set the LO offset, which is larger than the half of the signal bandwidth. The LO offset shifts the DC noise into the matched filter's stopband and therefore effectively suppresses the DC noise.

5.6.2 Bit error rates

We first evaluate the performance of different relaying strategies in terms of end-user bit-error rate (BER). Figure 5.6 illustrates the end-user raw and decoded BER of the TWRN with DNC, DF (with the MIMO zero-forcing receiver) and CF schemes, obtained during the experiments vs. the Tx gain of USRP. The end-user BER in TWRN is determined by uncorrected errors which occur during the MA phase and errors which occur during the broadcast phase. In our testbed the DNC and DF relays, however, apply the Adler-

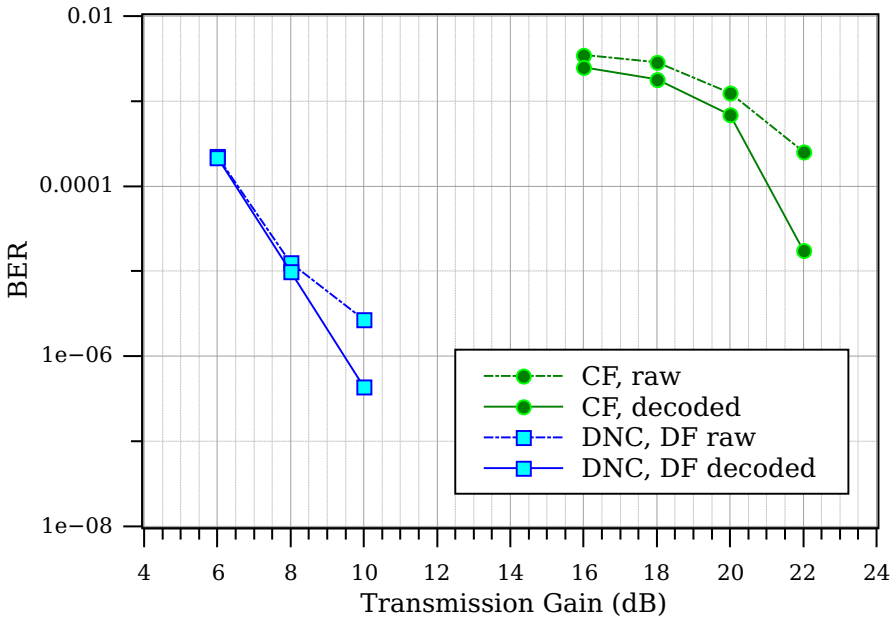
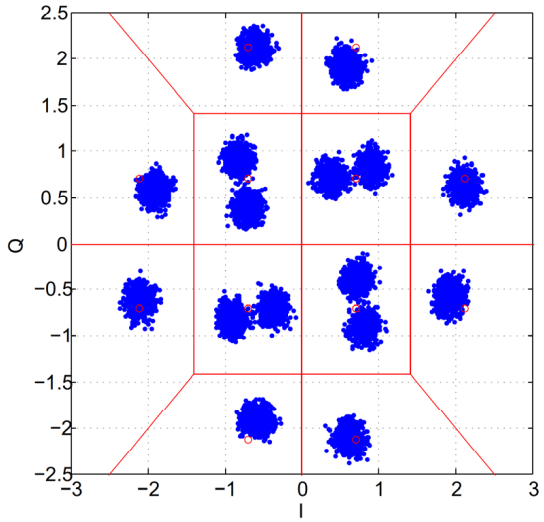
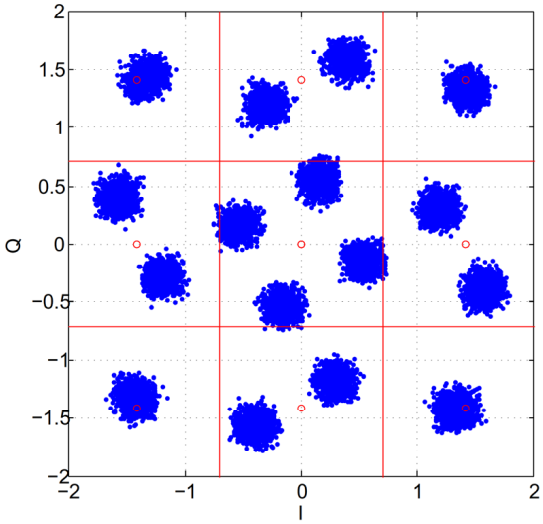


Figure 5.6: Measured BER comparison of TWRN with DNC, DF and CF vs. USRP Tx gain

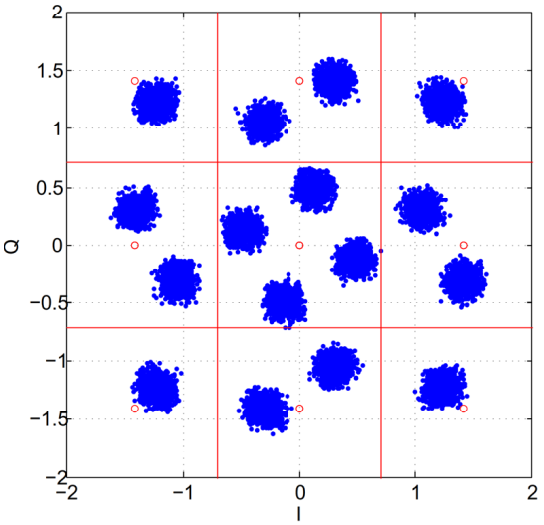
32 checksum algorithm for post-ECC data verification, and discard packets with errors uncorrected after ECC decoding. As a result, the BER of MA phase for those relaying schemes is negligible compared to the BER of the broadcast phase. Because the broadcast phase in the DF scheme is implemented in the same way as in the DNC scheme, these two schemes have similar BER. In fact, the decoded BER is similar to that of the RS code used for ECC. At the same time the CF relaying scheme exhibits considerably higher BER, both raw and decoded, for several reasons. First, the CF scheme suffers from the decrease of effective SNR, as demonstrated in Figure 5.7. Second, the CF scheme also suffers from imperfect symbol synchronization, which further increases the effective SNR. Finally, the Adler-32 algorithm is not implemented in the CF relay, because the relay does not recover bits from the received signals. Therefore, unlike the DNC and DF schemes, the CF relay is oblivious of ECC-uncorrected errors, hence corrupted packets are not discarded. Instead, the relay broadcasts the corrupted packets, which results in dramatic increase of the end-user BER. From the figure it is obvious, however, that the BER of the MA phase is partly improved by ECC decoding at terminals.



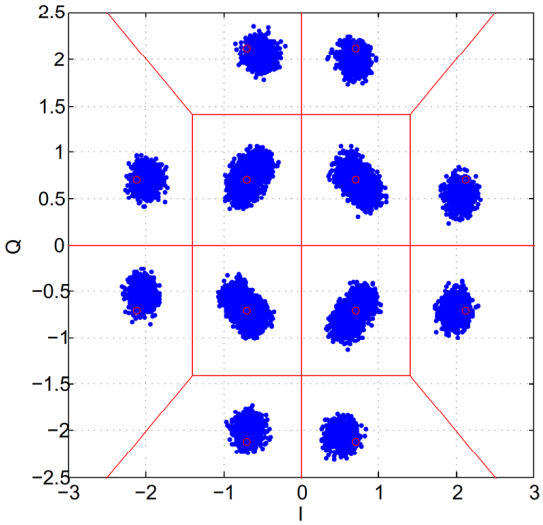
(a) $\mathbf{h} = [-0.148 + 0.021i, 0.039 - 0.080i]$, $\alpha_{\text{MMSE}} = 5.160 + 7.732i$, $\mathbf{a} = [-1 - i, 1]$



(b) $\mathbf{h} = [0.021 - 0.133i, 0.007 + 0.068i]$, $\alpha_{\text{MMSE}} = -0.572 - 8.075i$, $\mathbf{a} = [-1, 1]$



(c) $\mathbf{h} = [0.135 - 0.069i, -0.064 - 0.064i]$, $\alpha_{\text{MMSE}} = -4.030 + 6.039i$, $\mathbf{a} = [i, 1]$



(d) $\mathbf{h} = [-0.146 - 0.034i, 0.074 - 0.041i]$, $\alpha_{\text{MMSE}} = 8.057 + 4.794i$, $\mathbf{a} = [-1 - i, 1]$

Figure 5.7: Examples of constellations of received superimposed signal multiplied by α_{MMSE}, $\alpha_{\text{MMSE}}\mathbf{y}$, with different integer coefficients $\mathbf{a}$ which best fit the channel realization $\mathbf{h}$. The terminals transmit QPSK modulated signals. From $\alpha_{\text{MMSE}}\mathbf{y}$ the CF relay recovers linear combinations $\mathbf{x}_R$. The Voronoi diagrams represent decision regions of the slider (3.16) for each case.

Figure 5.7 presents several examples of constellations of superimposed signal $\mathbf{y}$ received by the relay, multiplied by the coefficient α with different integer coefficients $\mathbf{a}$

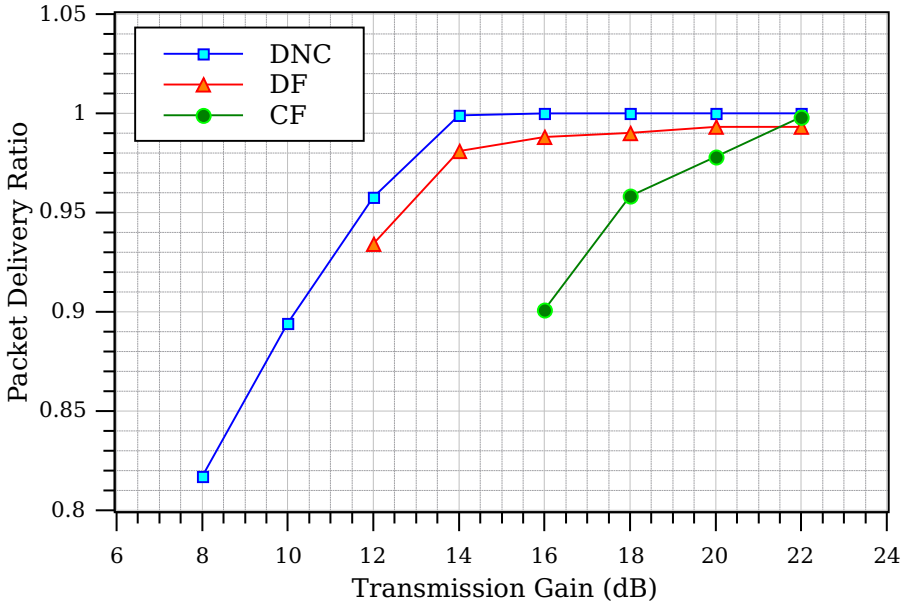


Figure 5.8: Measured PDR of TWRN with DNC, DF and CF relaying strategies vs. USRP Tx gain

which best fit the current channel state i.e. realizations of **h** and the SNR. Each subfigure demonstrates the relative amount of CF effective noise compared to the noise introduced by the channel.

5.6.3 Packet delivery ratio

Figure 5.8 shows the packet delivery ratio (PDR) of the TWRN with DNC, DF (with the MIMO zero-forcing receiver) and CF schemes, obtained during the experiments vs. the Tx gain of the USRP. As expected, the DNC strategy outperforms the other strategies in terms of PDR with any level of the Tx gain. When the Tx gain is higher than 14 dB, the PDR of the DF scheme is similar to that of the DNC strategy, although a performance gap is visible. This gap is determined by the imperfect synchronization typical for the GNU Radio platform and independent of the power level. The CF strategy shows limited performance with lower Tx gain. However, the CF's PDR becomes comparable to the performance of the DF strategy with increased Tx gain.

A comparison of these results with those provided in Figure 3.5 shows that the PDR

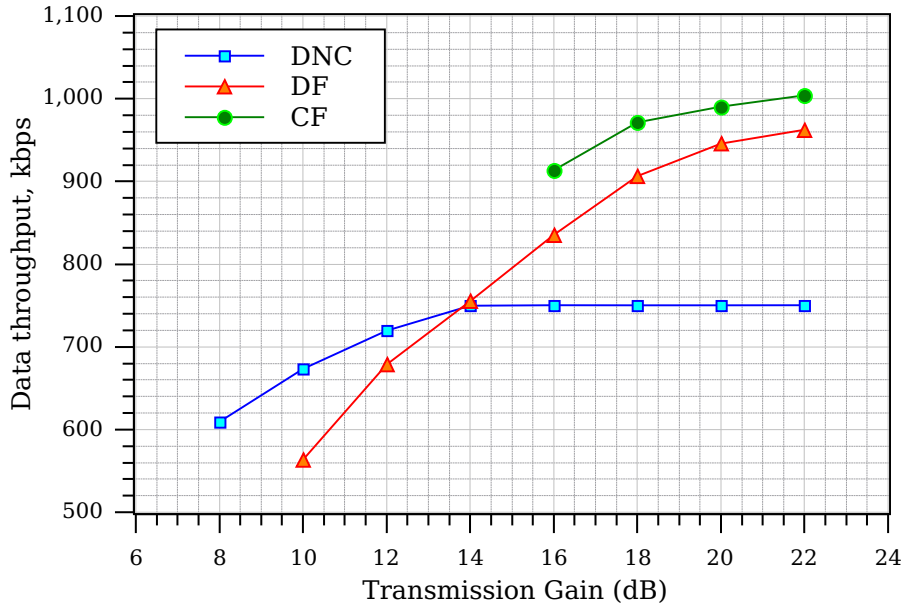


Figure 5.9: Measured data throughput per direction of TWRN with DNC, DF and CF relaying strategies vs. USRP Tx gain

of the DF scheme is less than that of the DNC scheme, while the FER of the DNC scheme is higher than that of the DF scheme. In the simulations in Section 3.7 we, however, assumed that the synchronization is perfect; this assumption is not achievable in practice. Therefore, even small errors in symbol synchronization negatively affect the actual PDR of the DF scheme. On the other hand, since the DNC relaying does not require synchronization, its performance remains similar.

5.6.4 Network throughput

Figure 5.9 illustrates the measured average data throughput (i.e., not counting the ECC redundancy and overhead) of the TWRN per direction vs. the Tx gain. With the Tx gain of 22 dB the average data throughput per direction is about 1000 kbps, 960 kbps and 750 kbps for the CF, DF and DNC schemes, respectively. The figure shows that with high Tx gain, the CF strategy outperforms the DNC by only about 33%, therefore, the theoretical advantage of 50% is not achieved. This is probably due to the presence of the synchronization delays and the need for additional retransmissions introduced by the TO-ARQ

protocol of the CF relay. However, this gap, could be partially reduced with a faster link between the host PC and USRP. Nevertheless, this figure illustrates that the CF scheme is beneficial in terms of the network throughput when the SNR is high.

We also observe that when the Tx gain is more than 16 dB, the DF TWRN throughput is less than that of the CF scheme. Furthermore, it also does not achieve the 50% improvement over the DNC scheme. This experimental result contradicts the simulation results illustrated in Figures 3.6 and 4.9, where the DF scheme had higher throughput at any SNR. The contradiction can be explained by the almost double amount of computations required in the DF relay, compared to that of the CF relay, because the DF relay processes signals from two antennas, and demodulates both packets individually. Therefore, this experiment demonstrates the computational efficiency of the CF scheme, when the scheme is implemented on an SDR platform. Note that in this experiment we implemented the MIMO zero-forcing receiver in the DF relay. Taking into account that the throughput decline occurs primarily due to the computational delays and hardware issues, we expect that the use of another MIMO receiver such as the MMSE receiver will not significantly improve performance of the DF scheme.

Finally, the figure shows that with lower Tx power the DNC relaying scheme provides higher throughput than both the CF and DF schemes. For example, when the Tx gain is 12 dB, the throughput of the DNC scheme is about 720 kbps, while the throughput of the DF scheme is only 680 kbps. At the same time, when the Tx gain was set to 8 dB, only the DNC scheme could provide reliable relaying with the throughput of about 600 kbps. Therefore, in this part the experimental results are consistent with those obtained in simulations shown in Figures 3.6 and 4.9.

5.7 Conclusion

In this chapter we have presented an implementation of a two-way relay network testbed with physical-layer network coding, namely the compute-and-forward relaying scheme. We have solved several problems which have hindered the development of physical-layer network coding in GNU Radio, such as hardware imperfections, software delays,

lack of synchronization and symbol-timing recovery for superimposed signals. As a result, compared to previous work, our testbed allows the implementation of standard PNC algorithms that require full synchronization.

Next, using the testbed, we have conducted several experiments and compared the performance of the compute-and-forward scheme with other relaying strategies such as digital network coding and decode-and-forward schemes. We have shown that in the high SNR regime, the compute-and-forward approach outperforms other relaying methods in terms of the network throughput. At the same time, the compute-and-forward relays require neither MIMO capabilities nor redundant computations for complete recovery of both packets, unlike the DF relay. Therefore, our experiments demonstrate the practical effectiveness of compute-and-forward relaying.

Further, in general the software-defined radio platform allows modification of the signal processing and relaying algorithms without additional costs. Therefore, the testbed and synchronization schemes which we have developed can serve as a base for further extension to more complex network coding scenarios within larger networks, both synchronous and asynchronous. The development of a larger network testbed and the investigation of physical-layer network coding efficiency in such networks are subjects of our future work.

Chapter 6

Multiplierless IIR filter design via zero/pole approximation

The pulse-shaping filter is an essential component of a communication system, used to limit the transmission bandwidth and minimize the intersymbol interference. Baseband pulse-shaping filters are typically implemented as a finite impulse response (FIR) filter. However, FIR filters are usually more computationally intensive than equivalent infinite impulse response (IIR) filters. We propose a new method for the design of multiplierless IIR pulse-shaping filters. First, a prototype floating-point IIR filter is designed using standard techniques (e.g. optimal Hankel-norm approximation). Then, using the proposed method, the filter's coefficients are converted into the canonical signed digit (CSD) form, which enables us to avoid using full multiplication operations. This method, based on zero/pole approximation, introduces a minor deviation of zeros and poles of the prototype filter, thus preserving the filter stability. The experimental results show that the bit-error rate introduced by a CSD IIR pulse-shaping filter is similar to that of an equivalent filter with FIR architecture. At the same time, the hardware complexity is significantly lower and the out-of-band power is reduced. This makes our CSD IIR filters particularly suitable for implementation on FPGAs.

6.1 Introduction

BASEBAND pulse-shaping filters for wireless communications have been used for decades, but their effective implementation on resource-constrained embedded systems remains a challenge. Various applications and different hardware implementation options have led to a wide variety of pulse-shaping filter implementations. Exam-

ples of practical implementations for a specific application, such as for DVB, WCDMA and underwater communications, have recently been proposed [129–133].

A common goal in all implementations is to reduce the number of multipliers or even to remove them completely. One method of multiplierless filter design is based upon the representation of filter coefficients as a sum of a small number of powers of two, referred to as canonical signed digit representation (CSD) [134]. In this case, multiplication by a coefficient can be implemented using only a small number of adders and shifters. Such filter design for finite impulse response (FIR) filters has received considerable attention [135–140].

FIR filters have many attractive properties, such as stability, linear phase, and robustness to coefficient quantization, but require a large number of multipliers. On the other hand, infinite impulse response (IIR) filters generally satisfy the desired frequency response specifications with lower order than that of an FIR filter. Therefore, despite the drawbacks, such as nonlinear phase and sensitivity to coefficient quantization, IIR filters can be a good alternative for reducing baseband receiver complexity and saving chip area. Various DSP techniques have been developed to minimize the drawbacks of IIR filters. For example, an IIR filter implemented as a cascade of second-order sections is more robust against coefficient quantization [141].

The design of IIR filters with CSD coefficients has attracted much less attention. In [142] Oh *et al.* proposed a method of designing a digital IIR filter with CSD coefficients based on solving the mixed integer linear programming problem. In [143], Liang *et al.* used a genetic algorithm to design an IIR filter as a cascade of second-order sections. The stability of each section was ensured by keeping the coefficients within the stability triangle. In [144], the design of 2D FIR and IIR filters with CSD coefficients using a genetic algorithm was discussed. A more recent paper [140] suggests a design algorithm based on solving an optimization problem, where the filter complexity is to be optimized. The filter complexity is defined as a number of nonzero digits in the CSD representation of all the coefficients. The complexity minimization is carried out via sequential coefficient truncation, subject to the constraints on the filter design requirements.

In this project we propose a different approach to CSD IIR filter design. Unlike other

methods based on optimized coefficient truncation, our approach is based on zero/pole approximation. This is based on the principle that a slight error in the location of poles does not affect the filter characteristics. Furthermore, we can ensure that the poles are still located within the unit circle in the z-plane, and therefore guarantee that the filter remains stable. As an example, we apply the proposed method to the root-raised cosine pulse-shaping filter design. We show that a CSD IIR pulse-shaping filter can be implemented with reduced hardware cost compared to those implemented as a CSD FIR filter. At the same time, the CSD IIR filter offers bit-error rates (BER) similar to those of the CSD FIR filter and provides lower out-of-band power.

The rest of this chapter is organized as follows. Section 6.2 introduces the proposed method of zero/pole approximation. Section 6.3 provides an example of a CSD IIR pulse-shaping filter design and its performance analysis. In Sections 6.4 and 6.5 the implementation and experimental results are discussed, and Section 6.6 summarizes our contribution to this part of the project.

6.2 The CSD zero/pole approximation method

In this section we review existing methods of IIR filter design as well as the CSD representation of rational numbers, and then describe the proposed algorithm of zero-pole approximation.

6.2.1 IIR filter design methods

A large number of methods for IIR filter design have been proposed. One group of methods, so-called direct methods, such as impulse invariance and bilinear transform, are based on conversion from analog filter prototypes. However, these methods are not always usable, for example in the case where the analytic form of the transfer function in the s-plane does not exist. Another group of methods is based on approximation of FIR prototype filters. This group includes the Pade approximation, the Chebyshev approximation and the optimal Hankel-norm approximation methods [145]. Others such as [146]

suggest the use of least-square approximation of an FIR filter by an IIR filter.

6.2.2 CSD representation

A rational number x is CSD representable with at most L nonzero digits if it represents a signed sum of powers of two:

$$x = \sum_{i=-M}^{N} \hat{s}_i 2^i = \sum_{n=1}^{L} s_n 2^{-p_n}, \tag{6.1}$$

where $\hat{s}_i \in \{-1, 0, 1\}$, $s_n \in \{-1, 1\}$ and $p_n \in \{-N, -N+1, \ldots, M\}$. Here $N+1$ is the total number of digits in the representation of the integer part and M is the total number of digits in the representation of the fractional part. We fix $N = 0$, (i.e., $|x| \leq \sum_{n=0}^{L-1} 2^{-n}$ if $L \leq M$) and denote the set of all CSD-representable numbers with a given L and M as Q_M^L. Obviously, $Q_M^{L-1} \subset Q_M^L$.

6.2.3 The algorithm

Let R_M^L denote the set of all possible roots in the complex plane of the quadratic equation $ax^2 + bx + c = 0$, when the coefficients have a CSD representation $a, b, c \in Q_M^L$ with specified L and M. Similarly, we let $R_M'^{L'}$ denote the set of all possible roots of the monic quadratic equation $x^2 + bx + c = 0$, where $b, c \in Q_M^{L'}$. An example of such a set, $R_7'^3$, is illustrated in Figure 6.1. Obviously, the number of R_M^L and $R_M'^{L'}$ increase with increasing of L and L', respectively, and in the complex plain they appear more densely distributed. We will use these two sets to approximate the zeros and poles on the desired transfer function.

Given an IIR filter with infinite precision coefficients, we can write the transfer function as the product of N_s second-order sections

$$H(z) \propto \prod_{k=1}^{N_s} \frac{(1 - \beta_{k1} z^{-1})(1 - \beta_{k2} z^{-1})}{(1 - \alpha_{k1} z^{-1})(1 - \alpha_{k2} z^{-1})}. \tag{6.2}$$

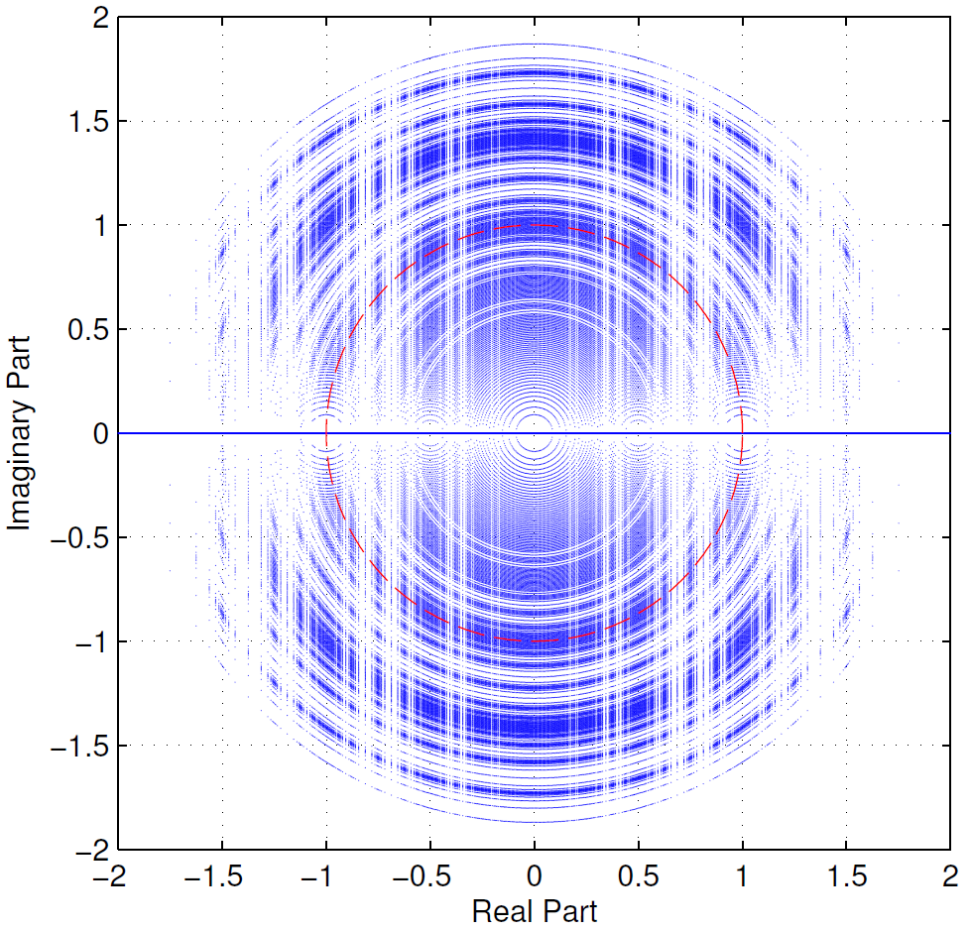


Figure 6.1: $R_7'^3$, set of all possible roots of equation $x^2 + bx + c = 0$, where $b, c \in Q_7^3$. The red circle represents the unit circle.

Each section has two zeros β_{k1} and β_{k2} and two poles α_{k1} and α_{k2} represented in infinite precision. The filter is stable if all the poles are located inside the unit circle in the z-plane, i.e., $|\alpha_{k1}| < 1$, $|\alpha_{k2}| < 1$.

Our goal is to design a filter with CSD coefficient representation and a transfer function:

$$\hat{H}(z) = \prod_{k=1}^{N_s} \hat{g}_k \frac{\hat{b}_{0k} + \hat{b}_{1k} z^{-1} + \hat{b}_{2k} z^{-2}}{1 + \hat{a}_{1k} z^{-1} + \hat{a}_{2k} z^{-2}} \propto \prod_{k=1}^{N_s} \frac{(1 - \hat{\beta}_{k1} z^{-1})(1 - \hat{\beta}_{k2} z^{-1})}{(1 - \hat{\alpha}_{k1} z^{-1})(1 - \hat{\alpha}_{k2} z^{-1})}, \tag{6.3}$$

where $\hat{b}_{0k}, \hat{b}_{1k}, \hat{b}_{2k} \in Q_M^L$ and $\hat{a}_{1k}, \hat{a}_{2k} \in Q_M^{L'}$. The corresponding zeros and poles are $\hat{\beta}_{k1}, \hat{\beta}_{k2} \in R_M^L$ and $\hat{\alpha}_{k1}, \hat{\alpha}_{k2} \in R_M'^{L'}$ and we select them to approximate the infinite precision

ones as

$$\min_{\hat{\beta}_{k1},\hat{\beta}_{k2}} |\hat{\beta}_{k1} - \beta_{k1}| + |\hat{\beta}_{k2} - \beta_{k2}|, \tag{6.4}$$

and

$$\min_{\substack{|\hat{\alpha}_{k1}|<1 \\ |\hat{\alpha}_{k2}|<1}} |\hat{\alpha}_{k1} - \alpha_{k1}| + |\hat{\alpha}_{k2} - \alpha_{k2}|. \tag{6.5}$$

Once the minimizing $\hat{\alpha}$'s $\hat{\beta}$'s are found, the corresponding $\hat{a}$ and $\hat{b}$ CSD coefficients are used for the filter implementation. Here $\hat{g}_k$ is the k-th section's scaling factor, which can be conveniently adjusted to a power of two in order to avoid overflow and underflow. Note that the minimization is carried out independently for each section's numerator and denominator and only needs to exhaustively search all the elements of R_M^L and $R_M'^{L'}$ respectively.

In summary, we have designed a filter with CSD coefficients with zeros and poles as close as possible to the zeros and poles of the original floating-point filter. This slight change of their locations marginally affects the filter characteristics. Experimental results in the next sections show that a very good approximation is achievable when the coefficient CSD representation contains at least three nonzero digits i.e. $L \geqslant 3$ and $L' \geqslant 3$.

6.3 Performance analysis

In this section, we present an example of an IIR root-raised cosine (RRC) filter with CSD coefficients implemented as a cascade of second-order sections. The impulse response of the ideal discrete-time RRC filter is

$$h_{RRC}(kT_s) = \begin{cases} \frac{1}{\sqrt{T}}\left(1-\rho+4\frac{\rho}{\pi}\right), & k=0, \\ \frac{\rho}{\sqrt{2T}}\left[\left(1+\frac{2}{\pi}\right)\sin\left(\frac{\pi}{4\rho}\right)+\left(1-\frac{2}{\pi}\right)\cos\left(\frac{\pi}{4\rho}\right)\right], & k=\pm\frac{T}{4\rho T_s}, \\ \frac{1}{\sqrt{T}}\dfrac{\sin\left[\pi\frac{kT_s}{T}(1-\rho)\right]+4\rho\frac{kT_s}{T}\cos\left[\pi\frac{kT_s}{T}(1+\rho)\right]}{\pi\frac{kT_s}{T}\left[1-\left(4\rho\frac{kT_s}{T}\right)^2\right]}, & \text{otherwise}, \end{cases} \tag{6.6}$$

where ρ is the roll-off factor, T is the symbol period, T_s is the sampling period, and $r \triangleq \frac{T}{T_s}$ is the oversampling factor. In this example we set $\rho = 0.3$ and $r = 8$.

As a performance benchmark, we consider an FIR root-raised cosine filter the coefficients of which are also in CSD form with at most three nonzero digits, i.e. from Q_M^3. We set $M = 15$, because the input signal resolution is 16 bits. The required minimum stopband attenuation is $A_{st} = 30$ dB. This FIR filter is implemented by rounding each floating-point coefficient of the root-raised cosine filter to the nearest CSD number. As a result of the rounding, the CSD FIR filter consists of 73 non-zero symmetric taps.

First, we design the prototype IIR filter with floating-point coefficients via the Hankel-norm approximation [145]. The desired stopband attenuation is achieved when the prototype IIR filter is composed of 6 second-order sections, which requires 30 multiplications per one input sample. Then, we convert the prototype filter in the CSD form with the proposed zero/pole approximation.

Figure 6.2 depicts the pole-zero plot of the original floating-point IIR filter, and the filters with CSD coefficients. According to the figure, the proposed approximation slightly shifts the zeros and poles from their original locations. However, since all the poles still lie within the unit circle, the CSD filter is stable. The values of the approximation errors for each pair of zeros and each pair of poles are provided in Table 6.1. From the table, the CSD representation of coefficients of the numerators with three nonzero digits, $L = 3$, approximates the zeros of the original filter with the worst-case error of the order of 10^{-4}. At the same time, the pole approximation errors are higher. In the case when $L' = 3$, the

Table 6.1: Zero/pole approximation error.

	Approximation Error		
k	Zeros, $L = 3$	Poles, $L' = 3$	Poles, $L' = 4$
1	$5.41 \cdot 10^{-6}$	$6.71 \cdot 10^{-3}$	$8.34 \cdot 10^{-4}$
2	$2.39 \cdot 10^{-4}$	$7.95 \cdot 10^{-3}$	$1.46 \cdot 10^{-3}$
3	$6.51 \cdot 10^{-4}$	$8.93 \cdot 10^{-3}$	$3.09 \cdot 10^{-3}$
4	$2.69 \cdot 10^{-5}$	$1.94 \cdot 10^{-2}$	$2.41 \cdot 10^{-3}$
5	$1.89 \cdot 10^{-4}$	$4.20 \cdot 10^{-3}$	$6.65 \cdot 10^{-4}$
6	$9.30 \cdot 10^{-5}$	$7.43 \cdot 10^{-3}$	$5.45 \cdot 10^{-4}$

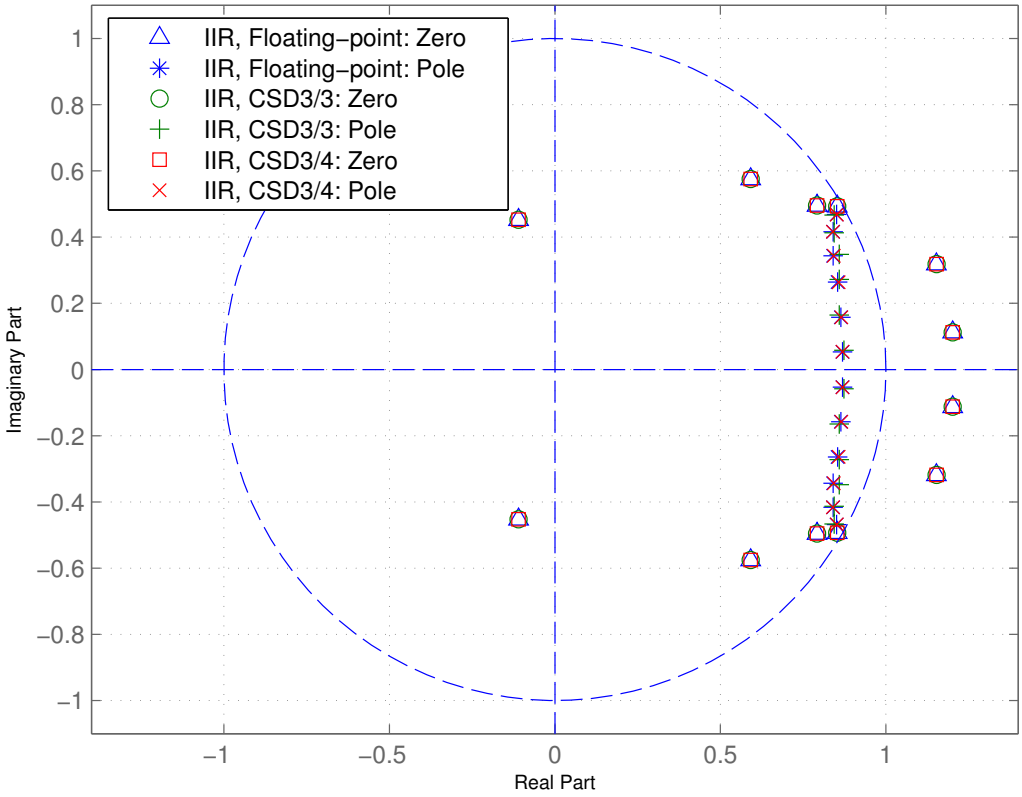


Figure 6.2: Pole-zero plot for the floating-point IIR filter, IIR filter with CSD3/3, and IIR filter with CSD3/4

largest error is equal to $1.94 \cdot 10^{-2}$. When $L' = 4$ the largest error is equal to $3.09 \cdot 10^{-3}$.

Figure 6.3 illustrates the frequency responses of the floating-point IIR filter and CSD IIR filters. When $L = 3$ and $L' = 4$, the frequency responses of the filters are similar. However, with $L' = 3$, the frequency response of the CSD filter exposes the undesired ripple in the passband. This ripple is due to the larger error of a pole approximation. In addition, the stopband attenuation of the CSD filter is insignificantly weaker. Figure 6.3 also compares the out-of-band power of the CSD IIR and CSD FIR filters. Although both the FIR and IIR filters satisfy the desired stopband attenuation, the IIR filter provides the stronger attenuation, thus reducing the out-of-band power.

In addition to the stopband attenuation requirement, another requirement for a pulse-shaping filter is minimization of the intersymbol interference (ISI). The zero-ISI criterion

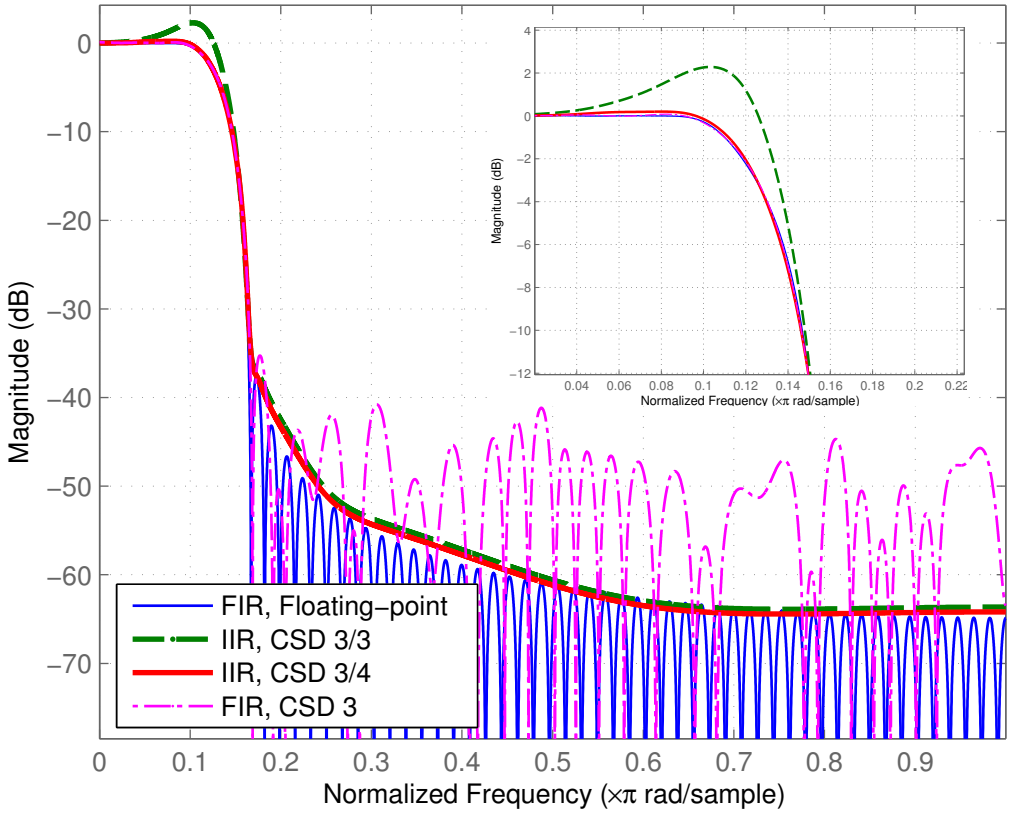


Figure 6.3: Frequency response of the floating-point FIR filter, CSD IIR filter with $L = L' = 3$, CSD IIR filter with $L = 3$, $L' = 4$, and CSD FIR filter with $L = 3$.

for a filter with impulse response $h_{PS}(t)$ is given as

$$h_{PS}(krT_s) = \begin{cases} 1, & k = 0, \\ 0, & k \neq 0. \end{cases} \tag{6.7}$$

As the raised cosine (RC) filter efficiently limits the bandwidth of the transmitted signal and its impulse response meets the zero-ISI criterion, the RC filter may be considered as a good pulse-shaping filter. However, due to the presence of white noise in the wireless channels, a matched filter is often employed in the receiver, in order to mitigate the effect of the white noise. In this scenario, it is more practical to split the RC impulse response between the transmitter and receiver, such that the combined impulse response $h(t)$ is

equal to the RC impulse response $h_{RC}(t)$, i.e.

$$h(t) = h_{Tx}(t) * h_{Rx}(t) = h_{RC}(t). \tag{6.8}$$

As $h_{RC}(t) = h_{RRC}(t) * h_{RRC}(t)$, the RRC filter can be used in both the transmitter and receiver

$$h_{Tx}(t) = h_{Rx}(t) = h_{RRC}(t). \tag{6.9}$$

Therefore, when designing the RRC filter with the proposed method we need to ensure that the combined impulse response $h(t)$ satisfies the zero-ISI criterion.

Figure 6.4 shows the normalized combined impulse responses $h(t)$ for different implementations of the root-raised cosine filters. In Figure 6.4(a) both the transmitter and receiver filters are implemented as the CSD IIR filter with $L = 3$ and $L' = 4$ via the proposed zero/pole approximation method, and in Figure 6.4(b) both the transmitter and receiver filters are implemented as the CSD IIR filter with $L = 3$ and $L' = 4$ via direct CSD truncation of the coefficients. The red solid markers show the impact of the inter-symbol interference (ISI) on consecutive impulses. From Figure 6.4(a), it follows that the zero-ISI requirement is not satisfied exactly, however the ISI introduced by the zero/pole approximation method is not significant. In fact, the ISI value between two consecutive pulses for CSD IIR filter is less than 2%. In contrast, Figure 6.4(b) demonstrates that the direct CSD truncation of the prototype filter's coefficients ruins the zero-ISI property.

The figure also illustrates the normalized impulse responses of the channel, comprised of the identical Tx and Rx floating-point IIR filters (Figure 6.4(c)). These impulse responses are compared to the channel response of the cascade of two ideal root-raised cosine filters, which correspond to the channel response of the ideal RC filter in Figures 6.4(d) to 6.4(f). From a comparison of Figure 6.4(d) and Figure 6.4(f) it follows that the higher ISI appears from the CSD IIR filter rather than from its floating-point prototype. Therefore, an ISI mitigation can be achieved by increasing L and L' for the sections where the accuracy of the zero/pole approximation is lower than in the other sections.

In summary, from the discussion above we see that three nonzero digits for the CSD representation of the coefficients of a second-order section in the numerator and four

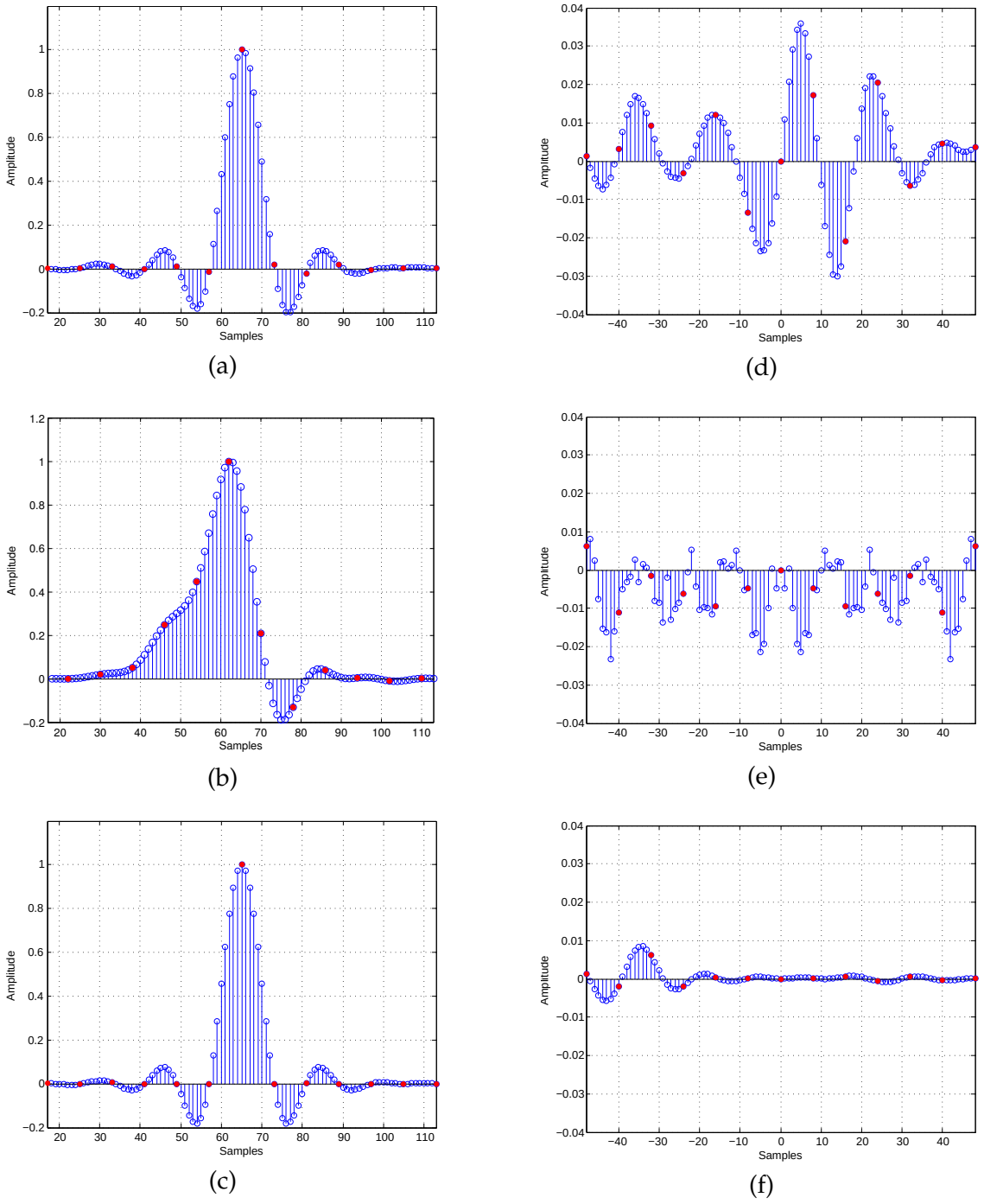


Figure 6.4: Normalized channel impulse response. (a) zero/pole approximation CSD IIR $L = 3$, $L' = 4$, (b) direct coefficient truncation CSD IIR $L = 3$, $L' = 4$, (c) Floating-point IIR; Difference between the RC impulse response and channel impulse response with (d) zero/pole approximation CSD IIR $L = 3$, $L' = 4$, (e) CSD3 FIR, (f) Floating-point IIR.

Table 6.2: Advanced HDL synthesis report. Macro statistics comparison for the FIR and IIR filters.

	FIR	IIR
Adders/Subtractors	16-bit adder : 193 16-bit subtractor: 100 Total: 293	20-bit adder : 86 20-bit subtractor : 106 Total: 192
Registers	Flip-Flops: 2858	Flip-Flops: 712

Table 6.3: Device utilization summary for the FIR and IIR filters.

		FIR		IIR	
Logic Utilization	Available	Used	%	Used	%
Number of Slice Flip-Flops	47744	2451	5	708	1
Number of 4 input LUTs	47744	4672	9	4002	8
Number of occupied Slices	23872	3580	14	2161	9
Total Number of 4 input LUTs	47744	4713	9	4002	8

nonzero digits in the denominator, $L = 3$ and $L' = 4$ are sufficient for practical CSD filter design. Such a filter has a similar frequency response to that of the floating-point prototype filter, and introduces tolerable ISI.

6.4 Hardware complexity

In this section, we compare the costs of hardware implementation of the CSD IIR filter with $L = 3$ and $L' = 4$, and the CSD FIR filter with $L = 3$ considered in the previous section. We synthesize the FPGA image for Xilinx Spartan 3A-DSP 3400 FPGA employed in the Ettus USRP N210 hardware platform.

Table 6.2 and Table 6.3 compare the FIR and IIR filter hardware requirements and actual hardware utilization respectively. From Table 6.3 it follows that the IIR filter is more hardware-efficient than the FIR filter. For example, the IIR filter occupies only 60% of slices and 85% of the four-input look-up tables (LUTs) occupied by the FIR filter. Hence, the proposed method of zero/pole approximation allows designing of the pulse-shaping

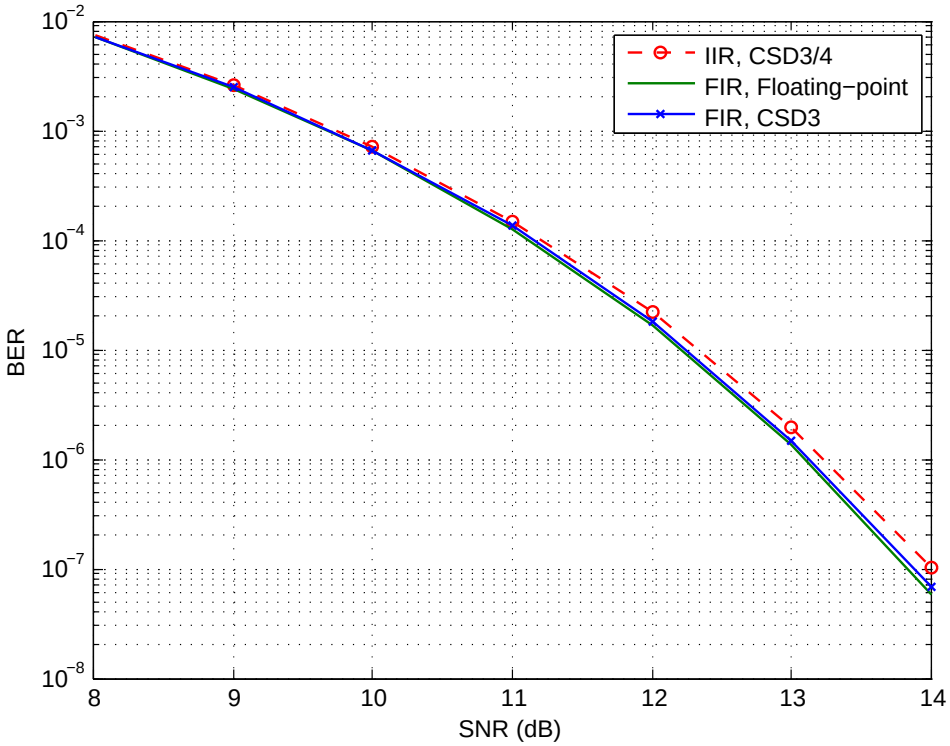


Figure 6.5: BER vs. SNR for the CSD FIR and CSD IIR pulse-shaping filters.

filter with reduced FPGA utilization.

6.5 BER performance

In order to verify the impact of the proposed pulse-shaping filter on the overall bit-error rate (BER) of the system, we performed simulations in GNU Radio. Figure 6.5 shows the BER vs. SNR curve for a software-defined communication system, where the transmitter and receiver filters are implemented as the CSD IIR filter (the red dashed line). The BER is compared with that of a system where both the transmitter and receiver filters are the root-raised cosine CSD FIR filter (the blue solid line). The BER of the floating point FIR system is also added to the figure (the green solid line). The type of modulation used is DQPSK. We see from the figure that the BER of the system with CSD IIR filters is marginally worse than the BER of the system with CSD FIR filters. This is probably due

to the fact that the pair of the IIR filters introduces a higher ISI than the pair of FIR filters. As a result, the hardware cost reduction is achieved along with a minor BER penalty.

6.6 Conclusion

We have proposed a new method for the design of IIR filters based on zero/pole approximation. The advantage of this method is that filter coefficients are represented in the CSD form, enabling efficient multiplierless implementation on FPGA. If CSD coefficients of a filter are represented by at least three nonzero digits, such a filter is stable, because the shift of the poles' locations is negligible. In addition, the frequency response of the filter is similar to that of the original IIR filter with floating-point coefficients.

Subsequently, we have applied the proposed method to the design of multiplierless IIR pulse-shaping filters. We show that a CSD IIR filter can be implemented with better stopband attenuation as well as reduced hardware complexity than those of a CSD FIR filter. However, the experimental results demonstrate that the combined impulse response of two such IIR filters has nearly zero-ISI, and provides a bit-error rate similar to that of the FIR filter.

Chapter 7
Conclusions and Future Work

7.1 Conclusions

Although PNC has attracted significant research attention in the last decade, the gap between theory and practice remains significant. The theoretical studies suggest that PNC has the potential to increase network throughput in some applications, such as wireless networks. At the same time, the practical use of PNC is burdened with several implementation challenges. As a result, only a few prototypes supporting PNC relaying have been designed. However, the CF relaying scheme, which demonstrates a number of advantages compared to other PNC methods, has not been implemented. In addition, some of those prototypes demonstrated low throughput when operating in real-time due to the lack of synchronization. Motivated by this fact and by improvements achieved with the use of digital network coding in various wireless networks, in this research we focused on prototyping efforts of the CF relaying scheme.

In this thesis we have designed the first real-time prototype of a TWRN with the CF relaying strategy and implemented it in GNU Radio. Using this testbed, we have conducted a number of experiments, which demonstrated that, with the proposed modifications, the CF scheme yielded a throughput improvement of about 30 % compared to that of the DNC relaying scheme. At the same time, the CF scheme did not require the use of multiple antennas at the relay.

In addition to the experimental demonstration of the potential of CF relaying, this testbed, implemented on the SDR platform, can be rapidly modified, extended and adapted for another PNC scheme. Therefore, this testbed can be used by the research commu-

nity for the design and evaluation of advanced PNC technologies, such as MIMO PNC, OFDM PNC, and beyond.

7.2 Summary of contributions

The following summarizes the detailed contributions of the research according to the objectives formulated in Section 1.3.

1. **Examining different PNC algorithms for their efficiency in scenarios typical for existing wireless communication standards and the simplicity of their implementation on SDR.**

   We began our research with an extensive literature review on PNC. We have found that the CF relaying scheme, in theory, outperforms other relaying strategies in terms of its BER. At the same time, this scheme can be easily deployed in multi-terminal, multi-relay networks, and the integration of the existing ECC with the CF relaying is straightforward. We have also analyzed the synchronization requirements for the synchronous PNC algorithms and the performance of asynchronous PNC algorithms. Our main focus was the simplicity of the implementation of asynchronous PNC algorithms. Our conclusion was that, while asynchronous PNC algorithms exhibited reasonable theoretical performance, their practical implementation presented a challenge with the existing hardware. In addition, we have overviewed some techniques which improve the quality of reception, such as channel estimation methods and the MAC-layer protocols, and their applicability in PNC. Overall, our review demonstrated that there was a gap between theoretical studies on PNC and their practical applicability to wireless communications.

2. **Development of a practical CF scheme based on realistic assumptions and suitable for use in actual communication systems.**

   In Chapter 3 we have developed a practical CF scheme which overcomes several draw-

backs of the original CF scheme, when it is utilized in actual wireless communications directly, without modification. In particular, the proposed scheme can make use of constellations with sizes equal to powers of two, without any performance degradation caused by the non-invertibility of the matrix of linear coefficients. At the same time, since the key components of this scheme are composed of very basic blocks, typically available in many SDR libraries, its prototyping is simple. As a result, the proposed CF scheme achieves the promised theoretical throughput. Our simulations demonstrated that, with the proposed modifications, the CF scheme outperformed other relaying strategies, such as the DNC scheme, when the SNR was high.

3. **Development of MAC protocols capable of supporting CF relaying and robust against GNU Radio inadequacies and USRP hardware imperfections.**

   In order to enhance the performance of the relaying schemes when they are implemented in GNU Radio, we have analyzed the challenges of implementation of packet-based communications in GNU Radio. Based on this analysis, we have developed a half-duplex packet-switching protocol which is efficient in SDR, particularly in GNU Radio. Using this protocol, we have designed and implemented the RT-ARQ protocol for the DNC and DF relaying schemes and the TO-ARQ protocol for the CF scheme. The main innovation is that the TO-ARQ protocol takes into account the fact that the CF relay is unable to recover the received packets individually. In addition, all the implemented ARQ protocols prevent unnecessary delays of the scheduler and enable the real-time operation of the testbed in GNU Radio.

4. **Implementation of a real-time prototype of TWRN with synchronous CF relaying strategy in GNU Radio.**

   In Chapter 5 we have demonstrated the first real-time testbed, where the relaying in TWRN was performed with the CF relaying strategy. We have analyzed several practical challenges which were hindering the development of PNC/CF relaying, and

proposed our solutions to minimize their negative impact. In particular, we designed a new synchronization scheme enabling both symbol and frame synchronization, and implemented this scheme on the FPGA of the USRP hardware. The proposed synchronization scheme allows synchronization sufficient for implementation of synchronous CF algorithms, which are more computationally efficient than non-synchronous algorithms. In addition, the synchronization between the terminals significantly simplifies the channel estimation, symbol-timing recovery and other routines compared to the asynchronous case. For the purpose of comparison, we also have implemented the DNC and DF relaying schemes on the same testbed.

5. **Experimental verification of the benefits of CF.**

   The availability of the testbed working in real-time gave us the opportunity to experimentally investigate the performance of the CF relaying scheme and compare it with other non-PNC relaying strategies. We demonstrated that when the SNR was high, the CF scheme suffered from higher FER and lower PDR than the DNC relaying strategy. Nevertheless, the CF scheme outperformed the DNC scheme in terms of throughput. However, the DNC scheme still provided better throughput in low-SNR regimes. Therefore, in general, our experimental results are in agreement with the simulation results. However, the most important contribution of our experiments is that, unlike the simulations provided in many previous studies, which only analyze the impact of fading and channel noise, our experiments evaluated the simultaneous impact of all the practical challenges. Therefore, the experimental results show that the practical benefits of the CF scheme still outweigh the negative impact of those challenges.

6. **Development of a new method of multiplierless pulse-shaping IIR filter design.**

   While working on the implementation of pulse-shaping filters on the USRP's FPGA, we found that the number of multipliers was limited and partially utilized by other modules of both the Tx and Rx chains. Therefore, the number of available multipliers

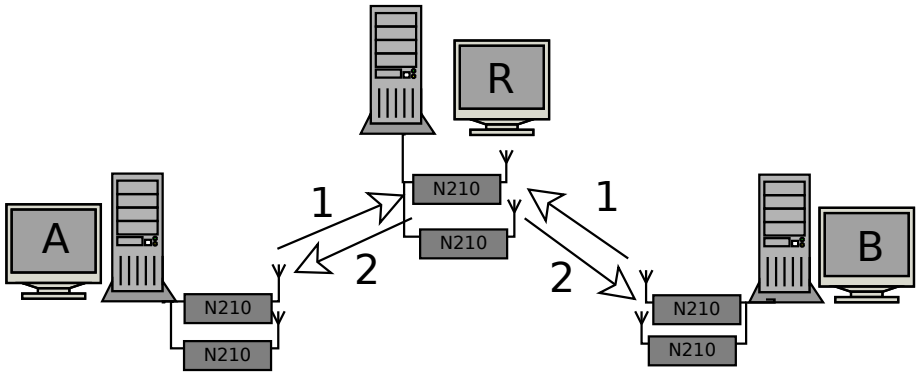


Figure 7.1: Two-way relay network with MIMO CF relaying implemented on USRP N210 and GNU Radio.

was insufficient for the pulse-shaping filter implementation. We therefore proposed a new method of multiplierless pulse-shaping filter design based on zero/pole approximation of a floating-point prototype IIR filter. The proposed method allows replacement of the conventional multiplications with those implemented using only adders and shifters, while still preserving the stability of the IIR filter. Our simulation results demonstrate that a filter with the desired characteristics can be implemented with reduced hardware utilization compared to that of the FIR filter prototype, but with a marginal increase of BER. Furthermore, the zero/pole approximation method enables implementation of digital filters on less expensive FPGAs which are not equipped with dedicated hardware multipliers.

7.3 Future work

There are several ways the research in this thesis can be continued in the future. These are summarized in the following sections.

7.3.1 Implementation of MIMO PNC/CF

MIMO PNC is another direction of study in PNC. In general, the use of MIMO in communication systems benefits with diversity gain, rate improvement and BER reduction.

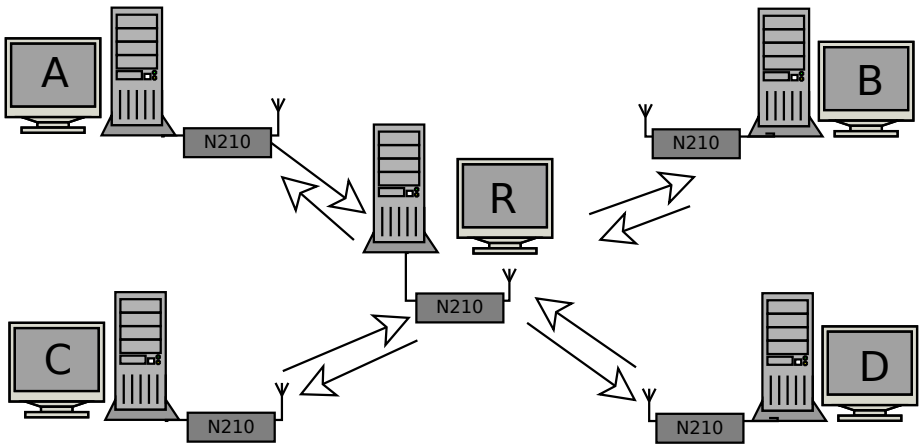


Figure 7.2: Relay network with the star topology and the CF relaying scheme implemented on USRP N210 and GNU Radio.

Recent theoretical studies on MIMO network coding are reviewed in [7, 17, 57]. A simplified implementation of TWRN with MIMO DF was also reported in [72]. With several implementation challenges solved, our TWRN testbed can be easily extended to support the MIMO PNC architecture, as illustrated in Figure 7.1. Subsequently, experimental performance analysis can be conducted in real environments, rather than simulations.

From the implementation point of view, the implementation of a MIMO terminal with two USRP N210s connected with the MIMO cable introduces extra delay, caused by the need to transfer data via the MIMO cable. As a result, the network throughput of the MIMO network may degrade greatly, making it impossible to accurately investigate the throughput performance improvements. Alternatively, PNC schemes in MIMO networks can be implemented with higher experimental efficiency, if the USRP N210 is replaced with a newer generation of USRP, such as X310 [147].

7.3.2 Extension of the testbed from TWRN to a larger network

In this project we have prototyped the CF relaying scheme in TWRN, a canonical example of a simple network topology with the relaying of bi-directional traffic. In general, however, the CF is designed such that it can be utilized in a larger network with multiple terminals. For example, a network with the star topology, as illustrated in Figure 7.2,

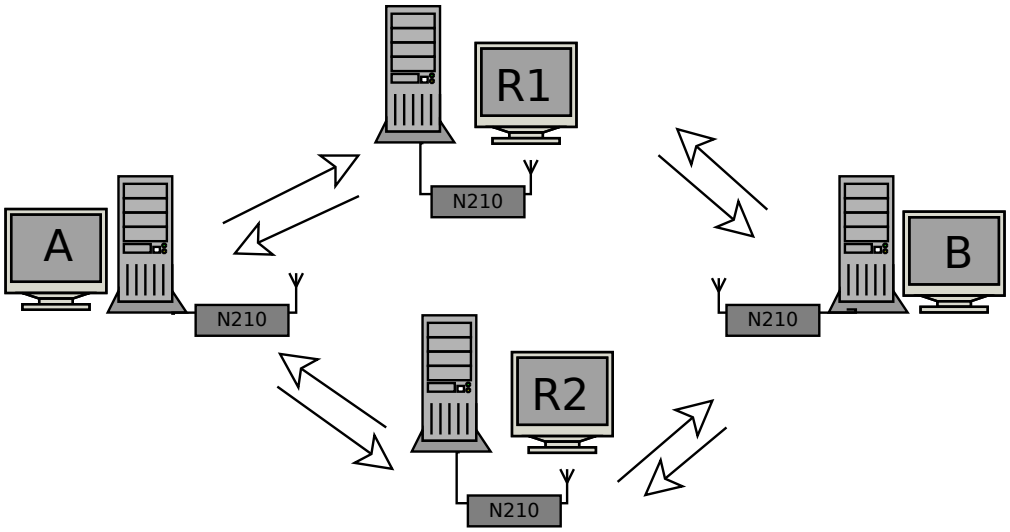


Figure 7.3: Relay network with multiple relays and the CF relaying scheme implemented on USRP N210 and GNU Radio.

assumes that there is a multi-directional traffic between more than two terminals. With this topology several terminals exchange information via only one relay. Another interesting extension is a multi-relay network where terminals transmit information via a group of relays, as presented in Figure 7.3. In this scenario, either one relay providing the best channel condition [148] or the maximum energy efficiency [149] can be selected for the relaying, or several relays can transmit simultaneously in order to increase transmit diversity [150].

7.3.3 Further optimization of zero/pole approximation method of CSD IIR filter design

In this work, we assumed that the prototype floating-point IIR filter was first designed from a FIR filter with the required specifications using any of the existing methods, and was stable. We then designed the CSD filter based on that prototype. This is suboptimal, because the design consists of two steps, and the optimization of each step is performed independently. A possible future research topic can be the development of a one-step method providing joint optimality for the design of the CSD IIR filter from the FIR filter prototype.

We have also assumed that the number of nonzero digits in the CSD coefficients representation was fixed. However, the accuracy of zero/pole approximation differs greatly in each section, so that for certain coefficients this number may be either redundant or insufficient. The optimization of the coefficient lengths with regard to the accuracy of the zero/pole approximation in each section can improve the performance of the CSD IIR filter without significant increase of hardware complexity. This optimization could be another possible future research topic.

Appendix A
Description of GRC flow graphs

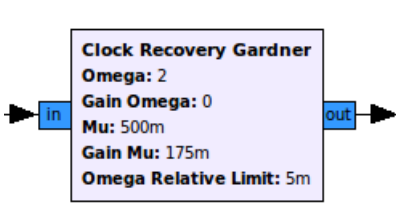
GNU Radio Companion (GRC) is a block diagram environment for modelling and simulating communication systems represented as a flow graph composed of blocks. GRC includes the graphical editor, the library of standard blocks (GNU Radio source tree) and the source code generator, which generates the Python script from a flow graph. The standard blocks are customizable, i.e. their functionality can be upgraded or rewritten completely depending upon the needs of the user. Also, users can create their own blocks, so-called *out-of-tree* (OOT) blocks. Development of OOT blocks is simplified with the *gr_modtool* utility provided in GNU Radio. In addition to the standard blocks and their own OOT blocks, users can include necessary OOT blocks from other third-party projects. In this appendix we first describe the OOT blocks we have designed for this project. Next, we present our GRC flow graphs which enable the operation of the terminals and relays in the DNC, DF and CF modes.

A.1 Design of custom blocks

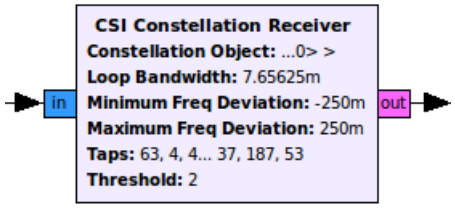
For this project we have designed several OOT blocks, the functionality of which was required in this testbed. These blocks include conventional PHY blocks, CF PHY blocks and half-duplex MAC blocks for the DNC, DF and CF relaying schemes as follows:

I. Conventional PHY blocks

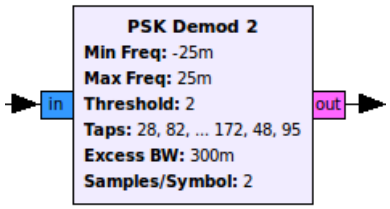
This group includes auxiliary blocks which implement several conventional (non-PNC) PHY routines not available in the source tree.



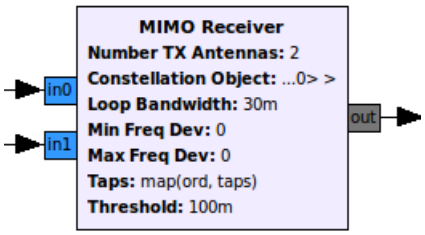
(a) Gardner Symbol-Timing Recovery block



(b) Coherent Receiver



(c) PSK Demod 2



(d) MIMO Receiver

Figure A.1: Conventional PHY OOT blocks.

1. **Gardner Symbol-Timing Recovery**. This block provides the same functionality as the **Clock Recovery MM** block in the Synchronizers library, i.e. symbol-timing recovery, with the exception that timing error detection is performed with the Gardner method. The main drawback of symbol-timing recovery with the MM method is that it is vulnerable to the frequency offset. Except that, the MM method is mainly designed to support PSK constellations (refer to Section 5.4 for detailed comparison). Therefore, despite the higher level of self-noise, the Gardner method seems a better solution for signals modulated with M-QAM constellations and superimposed constellations typical of PNC scenarios.

2. **Coherent Constellation Receiver**. This block is similar to the **Constellation Receiver** block from the source tree. We have added channel estimation functionality. This block searches for the start of a packet (SOP) based on correlation with a known training sequence. Once SOP is detected, the block performs channel estimation, compensates the channel effects, and demodulates the received packet.

3. **PSK Demod 2** This block provides demodulation of a PSK-modulated signal. The difference from the standard block **PSK Demod** is that two sub-blocks, namely

the AGC and RRC filter of **PSK Demod**, are removed. Therefore, this block allows the flow graph to process the input, supplied by a USRP with the customized FPGA image (refer to Section 5.3.3).

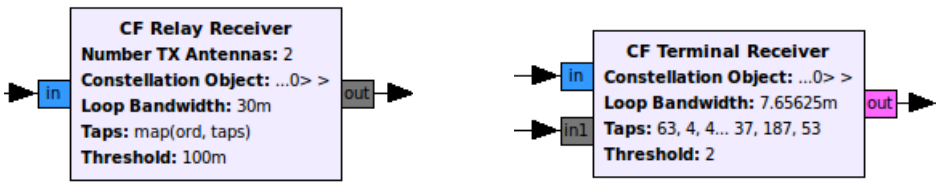
4. **MIMO Constellation Receiver**. This block represents an extension of the **Coherent Constellation Receiver** block for MIMO. This block searches for SOP, and once it is found, estimates the multiple channels and demodulates the received signals with the zero-forcing MIMO demodulation method.

As these blocks are nearly fixed-rate signal processing blocks, their implementation in C++ is straightforward.

II. CF blocks

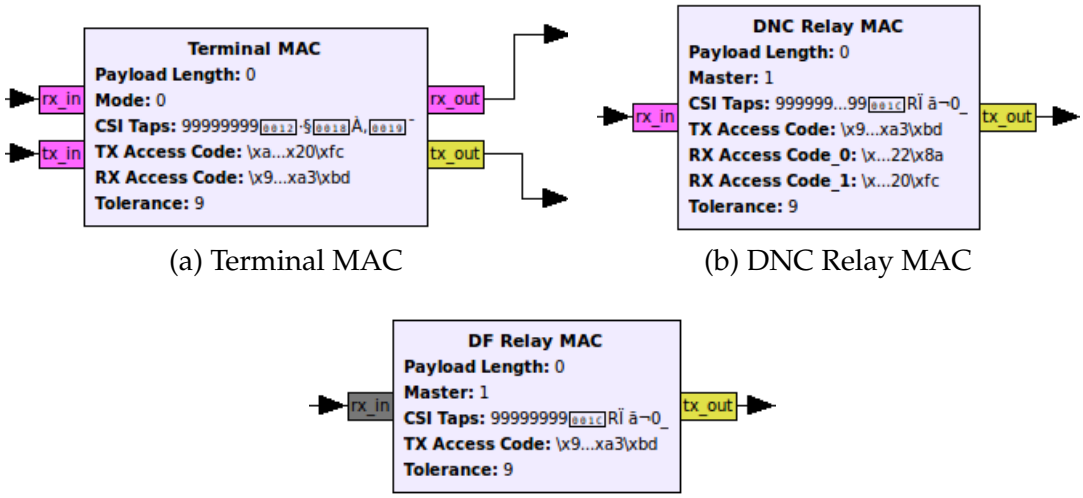
The second group includes PHY blocks required for implementation of the CF relay and CF terminals

1. **CF Relay Receiver**. This blocks represents the receiver of the CF relay. When receiving a superimposed packet, this block first searches for SOP based on the correlation with two known training sequences. Once the SOP is detected, the block estimates the channel coefficients **h**. Subsequently, it searches the optimal coefficients **a** according to equations (3.9) - (3.12), and recovers a linear combination of the superimposed signals as explained in equations (3.14) - (3.16). In addition, due to the impossibility of implementation of a dedicated half-duplex block in CF relay, this block coordinates the half-duplex functioning of the relay according to Figure 4.5(b).



(a) CF Relay Receiver (b) CF Terminal Receiver

Figure A.2: CF PHY OOT blocks.



(a) Terminal MAC

(b) DNC Relay MAC

(c) DF Relay MAC

Figure A.3: MAC OOT blocks.

2. **CF Terminal Receiver**. The terminal part of the CF algorithm proposed in Chapter 3 is implemented in this block, namely the routines explained in equations (3.17) - (3.21). The block provides recovery of the target packet ($\mathbf{m}_A$ or $\mathbf{m}_B$) from the linear combination $\hat{\mathbf{x}}_R$ broadcast by the relay. This block has two input ports. The linear combination $\hat{\mathbf{x}}_R$ is supplied via the first input port *in*, and the block's own transmitted signal ($\mathbf{x}_B$ or $\mathbf{x}_A$) is provided via the second input port *in1*. The type of port *in1* is chosen to be "message queue" in order to avoid errors due to the fact that GNU Radio prohibits the looping of blocks with connections of numeric types.

These blocks are also fixed-rate signal processing blocks implemented in C++.

III. Half-duplex MAC blocks

Finally, this group represents MAC blocks utilized in terminals, and DF and DNC relays for half-duplex packet switching

1. **Terminal MAC**. This half-duplex block is responsible for the MAC layer routines in DNC, DF and CF terminals according to Figure 4.2.

2. **DNC Relay MAC**. This block has expended functionality compared to the previous block. It performs MAC routines for the DNC relay according to Figure 4.5(a), i.e. verification of two packets arriving sequentially, and XOR network coding of successfully received packets. This block outputs the resulting XORed packet.

3. **DF Relay MAC**. This block has similar functionality to the previous block. It performs MAC routines for the DF relay according to Figure 4.5(a), i.e. verification of two packets arriving simultaneously, and XOR network coding of successfully received packets. This block outputs the resulting XORed packet.

Unlike the blocks from previous groups, these MAC blocks require extensive use of external libraries, such as the ZLIB library for the Adler-32 checksum algorithm. As these blocks are not fixed-rate blocks, and their output is represented by packets rather than continuous streams, they are implemented in Python. The type of port for *tx_out* port of all these blocks can be selected either as *byte* (8 bit) or as *short* (16 bit) depend on the structure of the Tx chain. If the modulation and pulse-shaping are performed in GNU Radio, the output type should be selected as *byte*. Otherwise, if the Tx chain is relocated to the USRP, the type of output should be *short*, and the block should be connected directly to a USRP Sink block.

In this section we have introduced the OOT blocks we have developed for the use in this testbed. This introduction simplifies the description of the testbed design flow graphs, provided in the following sections.

A.2 Relay design in GRC

The flow graphs which represent the relay design for different relaying strategies are illustrated in Figures A.4 to A.6.

The flow graph describing the relay operation in the DNC mode is presented in Figure A.4. The design of this flow graph is straightforward.

Figure A.5 illustrates the flow graph of the relay operating in the DF mode. In this

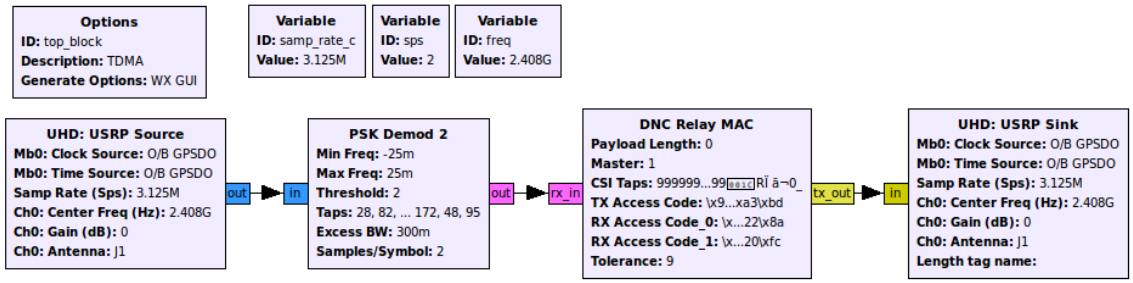


Figure A.4: GRC flow graph of the DNC relay.

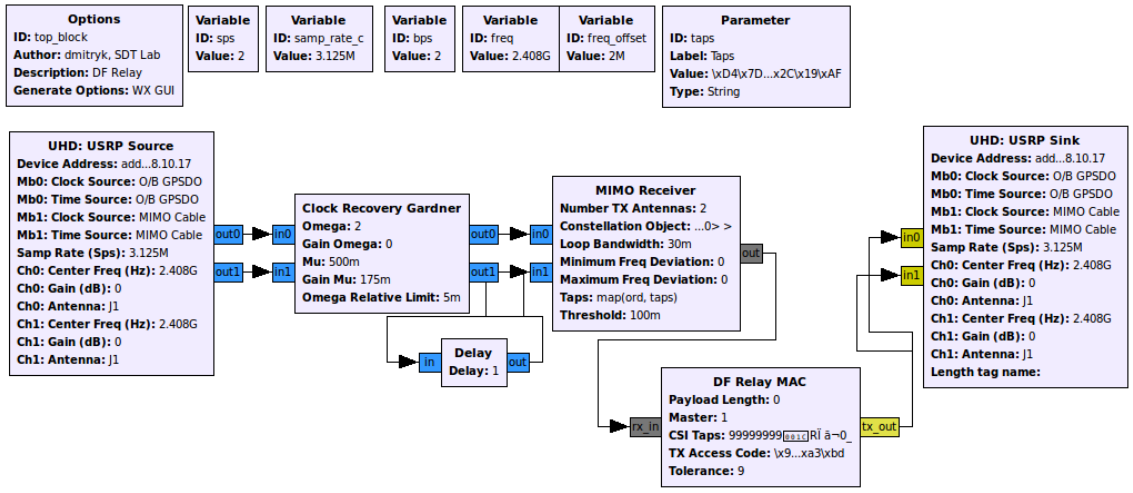


Figure A.5: GRC flow graph of the DF relay.

mode, the relay is a MIMO receiver, but a single antenna transmitter. Because the UHD does not allow two channels in the USRP Source and one channel in the USRP Sink, when they represent one device, we set the number of channels in the USRP Sink and USRP Source to two. However, the second input is multiplied by zero in the FPGA to avoid actual transmission by the second antenna.

Figure A.6 shows the flow graph of the relay operating in the CF mode. The CF relay has some special features from the GNU Radio implementation point of view. First, the CF relay does not decode the packets $\mathbf{m}_A$ and $\mathbf{m}_B$ individually. Second, unlike the DNC and DF schemes, the ECC decoding is performed jointly with demodulation, rather than after demodulation. Therefore, the half-duplex packet-switching logic and the ECC decoding/re-encoding are merged into one block.

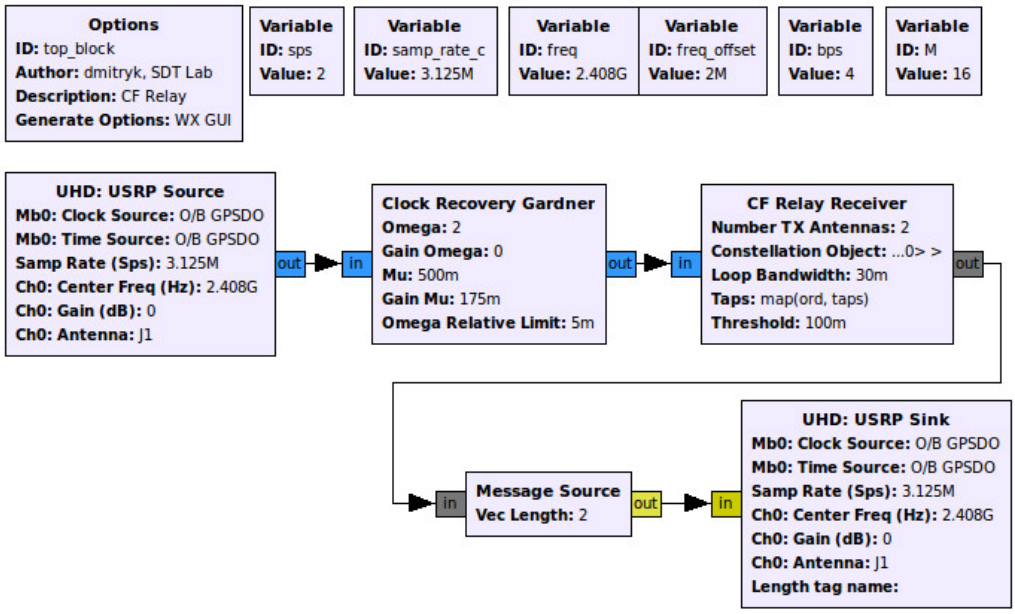


Figure A.6: GRC flow graph of the CF relay.

A.3 Terminal design in GRC

Unlike the relays, terminals deliver the data sent by the other terminal to the end-users or applications, or record the data. In this project we record the received data into files. For this reason, a File Sink block is added to every terminal. However, in other projects, the File Sink block can be replaced with other sinks such as audio, UDP or TCP sinks. If the received data is not actually required, it can be consumed with the Null Sink block. In the same way, the data to be transmitted is supplied to the flow graph with the File Source block.

The flow graphs of terminals are shown in Figures A.7 and A.8. Figure A.7 shows a flow graph of a terminal adapted for the DNC and DF schemes. The Skip Head block is used to skip the dummy data from the first packet sent for initialization. The network decoding is implemented with one input port of the XOR block connected to the output port *rx_out* of the Terminal MAC block through the Skip Head block. The second input port of the XOR block is connected to the File Source block. In this way, the XOR block performs recovery of the received packets according to (1.2). Note that due to the similar format of the broadcast phase in DNC and DF TWRN, a DNC and DF terminal can be

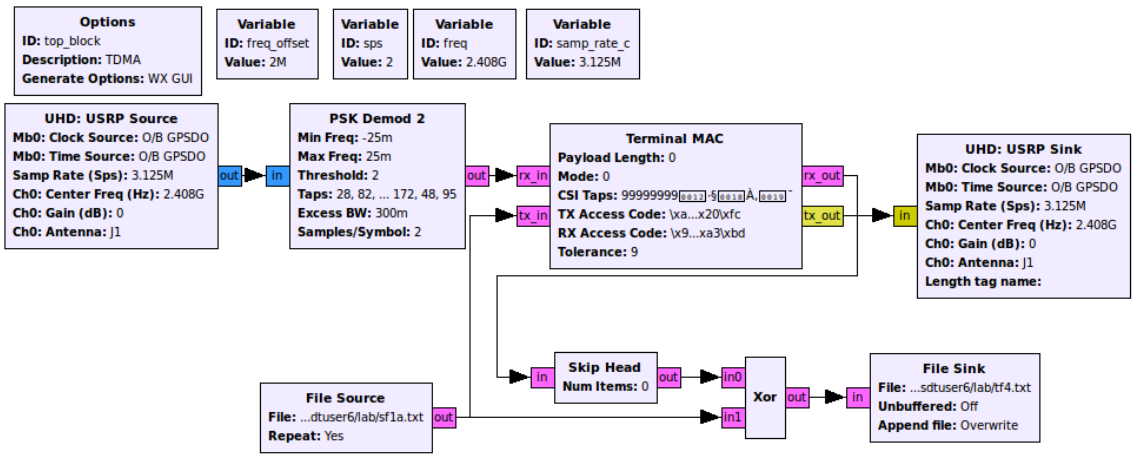


Figure A.7: GRC flow graph of the DNC and DF terminal.

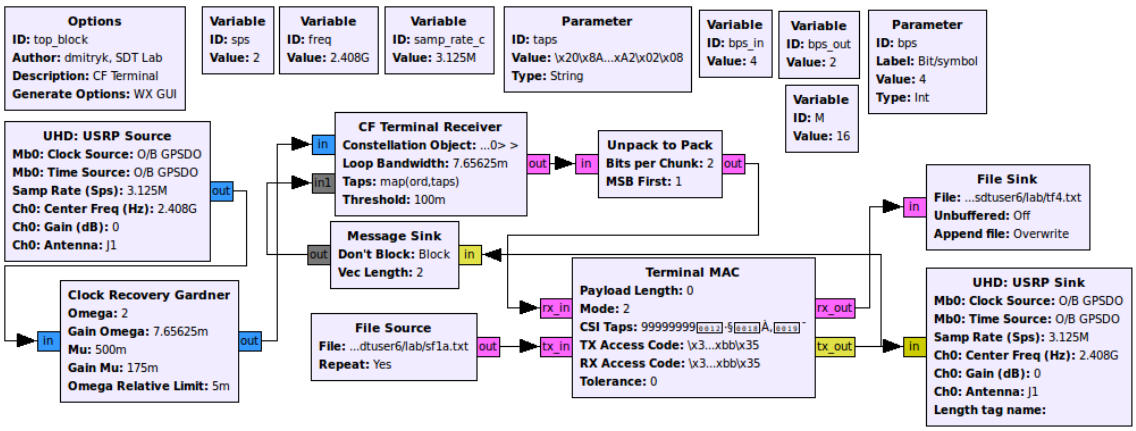


Figure A.8: GRC flow graph of the CF terminal.

implemented with the same flow graph with only a minor change related to the use of different channel estimation methods, depending on whether the packets arrive simultaneously (DF) or sequentially (DNC). This difference can be reflected with a change in the *CSI Taps* field.

Finally, the flow graph of the relay operating in the CF mode is demonstrated in Figure A.8. After the desired signal is recovered and demodulated by the CF Terminal Receiver block, the integrity of the packet is verified with the Terminal MAC block, which sends correctly received packets to the File Sink block. In turn, the Terminal MAC block prepares the Tx packet, which is sent to the USRP Sink and subsequently transmitted.

Bibliography

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, vol. 1, no. 2, pp. 41–59, Autumn 1996.

[2] G. G. Raleigh and J. M. Cioffi, "Spatio-temporal coding for wireless communication," *IEEE Transactions on Communications*, vol. 46, no. 3, pp. 357–366, Mar. 1998.

[3] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, Oct. 1998.

[4] J. C. Belfiore, G. Rekaya, and E. Viterbo, "The golden code: a 2×2 full-rate space-time code with nonvanishing determinants," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1432–1436, Apr. 2005.

[5] H. Yao and G. W. Wornell, "Achieving the full MIMO diversity-multiplexing frontier with rotation-based space-time codes," in *in Proc. Allerton Conf. Commun., Contr., Comput., IL*, 2003.

[6] A. Nosratinia, T. E. Hunter, and A. Hedayat, "Cooperative communication in wireless networks," *IEEE Communications Magazine*, vol. 42, no. 10, pp. 74–80, Oct. 2004.

[7] B. Nazer and M. Gastpar, "Reliable physical layer network coding," *Proceedings of the IEEE*, vol. 99, pp. 438–460, Mar. 2011.

[8] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[9] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *Information Theory, IEEE Transactions on*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[10] R. Koetter and M. Medard, "An algebraic approach to network coding," *Networking, IEEE/ACM Transactions on*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[11] Y. Wu, P. A. Chou, and S.-Y. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-2004-78, Aug. 2004. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=78174

[12] D. Lun, N. Ratnakar, M. Medard, R. Koetter, D. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2608–2623, June 2006.

[13] S. Zhang, Y. Zhu, and S. C. Liew, "Soft network coding in wireless two-way relay channels," *Communications and Networks, Journal of*, vol. 10, no. 4, pp. 371–383, Dec. 2008.

[14] S. Zhang, S. C. Liew, and P. P. Lam, "Hot topic: Physical-layer network coding," in *Proc. of the 12th Annual Int. Conf. on Mobile Computing and Networking (MobiCom)*. LA, USA: ACM, Sept. 2006, pp. 358–365.

[15] P. Popovski and H. Yomo, "The anti-packets can increase the achievable throughput of a wireless multi-hop network," in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 9, June 2006, pp. 3885–3890.

[16] B. Nazer and M. Gastpar, "Computing over multi-access channels with connections to wireless network coding," in *IEEE International Symposium on Information Theory (ISIT)*. Seattle, USA: IEEE, July 2006, pp. 1354–1358.

[17] S. C. Liew, S. Zhang, and L. Lu, "Physical-layer network coding: Tutorial, survey, and beyond," *Physical Communication*, vol. 6, pp. 4–42, Mar. 2013.

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

[19] H. Seferoglu, L. Keller, B. Cici, A. Le, and A. Markopoulou, "Cooperative video streaming on smartphones," in *49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE. Monticello, IL: IEEE, Sept. 2011, pp. 220–227.

[20] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou, "MicroCast: Cooperative video streaming on smartphones," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. New York, NY, USA: ACM, 2012, pp. 57–70.

[21] R. Alimi, L. Li, R. Ramjee, H. Viswanathan, and Y. Yang, "iPack: in-network packet mixing for high throughput wireless mesh networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, IEEE. IEEE, Apr. 2008.

[22] M. Firooz, Z. Chen, S. Roy, and H. Liu, "Wireless network coding via modified 802.11 MAC/PHY: Design and implementation on SDR," *Selected Areas in Communications, IEEE Journal on*, vol. 31, no. 8, pp. 1618–1628, Aug. 2013.

[23] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, Feb. 2014.

[24] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, June 2014.

[25] E. Lloyd and G. Xue, "Relay node placement in wireless sensor networks," *Computers, IEEE Transactions on*, vol. 56, no. 1, pp. 134–138, Jan. 2007.

[26] Z. Liu, M. Li, L. Lu, C.-K. Chan, S.-C. Liew, and L.-K. Chen, "Optical physical-layer network coding," *Photonics Technology Letters, IEEE*, vol. 24, no. 16, pp. 1424–1427, Aug. 2012.

[27] M. Li, Y. Wu, L.-K. Chen, and S. C. Liew, "Common-channel optical physical-layer network coding," *Photonics Technology Letters, IEEE*, vol. 26, no. 13, pp. 1340–1343, July 2014.

[28] "Ettus USRP N210." [Online]. Available: https://www.ettus.com/product/details/UN210-KIT

[29] T. Koike-Akino, P. Popovski, and V. Tarokh, "Denoising maps and constellations for wireless network coding in two-way relaying systems," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, Nov. 2008, pp. 1–5.

[30] S. Zhang, S. C. Liew, and L. Lu, "Physical layer network coding schemes over finite and infinite fields," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, Nov. 2008, pp. 1–6.

[31] T. Cui, T. Ho, and J. Kliewer, "Memoryless relay strategies for two-way relay channels," *Communications, IEEE Transactions on*, vol. 57, no. 10, pp. 3132–3143, Oct. 2009.

[32] S. Katti, S. Gollakota, and D. Katabi, "Embracing wireless interference: Analog network coding," in *SIGCOMM 2007*. Kyoto, Japan: SIGCOMM, Aug. 2007.

[33] T. Cover and A. Gamal, "Capacity theorems for the relay channel," *Information Theory, IEEE Transactions on*, vol. 25, no. 5, pp. 572–584, Sept. 1979.

[34] G. Kramer, M. Gastpar, and P. Gupta, "Cooperative strategies and capacity theorems for relay networks," *Information Theory, IEEE Transactions on*, vol. 51, no. 9, pp. 3037–3063, Sept. 2005.

[35] T. Koike-Akino, P. Popovski, and V. Tarokh, "Optimized constellations for two-way wireless relaying with physical network coding," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 5, pp. 773–787, June 2009.

[36] S. Wang, Q. Song, L. Guo, and A. Jamalipour, "Constellation mapping for physical-layer network coding with M-QAM modulation," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, Dec. 2012, pp. 4429–4434.

[37] Y. Huang, Q. Song, S. Wang, and A. Jamalipour, "Phase-level synchronization for physical-layer network coding," in *IEEE Global Communications Conference (GLOBECOM)*. Anaheim, CA: IEEE, Dec. 2012, pp. 4423 – 4428.

[38] R. Louie, Y. Li, and B. Vucetic, "Practical physical layer network coding for two-way relay channels: performance analysis and comparison," *Wireless Communications, IEEE Transactions on*, vol. 9, no. 2, pp. 764–777, Feb. 2010.

[39] P. Popovski and H. Yomo, "Physical network coding in two-way wireless relay channels," in *Communications, 2007. ICC '07. IEEE International Conference on*. Glasgow: IEEE, June 2007, pp. 707 – 712.

[40] S. Zhang and S.-C. Liew, "Channel coding and decoding in a relay system operated with physical-layer network coding," *Selected Areas in Communications, IEEE Journal on*, vol. 27, no. 5, pp. 788–796, June 2009.

[41] Z. Ding, I. Krikidis, J. Thompson, and K. Leung, "Physical layer network coding and precoding for the two-way relay channel in cellular systems," *Signal Processing, IEEE Transactions on*, vol. 59, no. 2, pp. 696–712, Feb. 2011.

[42] S. Zhang and S. C. Liew, "Physical layer network coding with multiple antennas," in *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, Apr. 2010, pp. 1–6.

[43] M. Huang and J. Yuan, "Error performance of physical-layer network coding in multiple-antenna TWRC," *Vehicular Technology, IEEE Transactions on*, vol. 63, no. 8, pp. 3750–3761, Oct. 2014.

[44] N. Lee and R. W. Heath, "Space-time physical-layer network coding," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 2, pp. 323–336, Feb. 2015.

[45] E. Biglieri, R. Calderbank, A. Constantinides, A. Goldsmith, A. Paulraj, and H. V. Poor, *MIMO Wireless Communications*. Cambridge university press, 2007.

[46] J. Zhan, B. Nazer, U. Erez, and M. Gastpar, "Integer-forcing linear receivers," *Information Theory, IEEE Transactions on*, vol. 60, no. 12, pp. 7661–7685, Dec. 2014.

[47] S. M. Azimi-Abarghouyi, M. Nasiri-Kenari, and B. Maham, "Integer forcing-and-forward transceiver design for MIMO multi-pair two-way relaying," 2015, submitted to IEEE Transactions on Vehicular Technology. [Online]. Available: http://arxiv.org/abs/1408.2854

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

[49] O. Ordentlich, J. Zhan, U. Erez, M. Gastpar, and B. Nazer, "Practical code design for compute-and-forward," in *IEEE ISIT 2011*. St.-Petersburg: IEEE, Aug. 2011, pp. 1876–1880.

[50] A. Sakzad, E. Viterbo, J. Boutros, and Y. Hong, "Phase precoded compute-and-forward with partial feedback," in *IEEE International Symposium on Information Theory (ISIT)*. Honolulu, HI: IEEE, June 2014, pp. 2117–2121.

[51] B. Hern and K. Narayanan, "Multilevel coding schemes for compute-and-forward with flexible decoding," *IEEE Transactions on Information Theory*, vol. 59, no. 11, pp. 7613–7631, Nov. 2013.

[52] U. Niesen and P. Whiting, "The degrees of freedom of compute-and-forward," *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5214–5232, Aug. 2012.

[53] C. Feng, D. Silva, and F. Kschischang, "An algebraic approach to physical-layer network coding," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, IEEE. Austin, TX: IEEE, June 2010, pp. 1017 – 1021.

[54] ——, "An algebraic approach to physical-layer network coding," *IEEE Transactions on Information Theory*, vol. 59, no. 11, pp. 7576 – 7596, Nov. 2013.

[55] Q. T. Sun, T. Huang, and J. Yuan, "On lattice-partition-based physical-layer network coding over GF(4)," *IEEE Communications Letters*, vol. 17, no. 10, pp. 1988–1991, Oct. 2013.

[56] T. Huang, J. Yuan, and J. Li, "Analysis of compute-and-forward with QPSK in two-way relay fading channels," in *2013 Australian Communications Theory Workshop (AusCTW)*, Adelaide, Australia, Feb. 2013, pp. 75 – 80.

[57] J. Zhan, B. Nazer, M. Gastpar, and U. Erez, "MIMO compute-and-forward," in *IEEE International Symposium on Information Theory (ISIT)*, IEEE. Seoul, Korea: IEEE, June 2009, pp. 2848–2852.

[58] S. Zhang, S. C. Liew, and P. P. Lam, "On the synchronization of physical-layer network coding," in *IEEE Information Theory Workshop (ITW)*. Chengdu, China: IEEE, Oct. 2006, pp. 404 – 408.

[59] L. Lu and S. C. Liew, "Asynchronous physical-layer network coding," *Wireless Communications, IEEE Transactions on*, vol. 11, no. 2, pp. 819–831, Feb. 2012.

[60] L. Lu, S. C. Liew, and S. Zhang, "Optimal decoding algorithm for asynchronous physical-layer network coding," in *IEEE ICC*. Kyoto, Japan: IEEE, June 2011, pp. 1–6.

[61] ——, "Channel-coded collision resolution by exploiting symbol misalignment," in *Communications (ICC), 2010 IEEE International Conference on*, Cape Town, May 2010, pp. 1–6.

[62] H. Najafi, M. O. Damen, and A. Hjrungnes, "Symbol-asynchronous compute-and-forward," in *IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications*, Toronto, ON, Sept. 2011, pp. 1830 – 1834.

[63] ——, "Asynchronous compute-and-forward," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2704–2712, July 2013.

[64] Y. Li, F.-C. Zheng, and M. Fitch, "Physical layer network coding with channel and delay estimation," *Communications, IET*, vol. 7, no. 11, pp. 1109–1116, July 2013.

[65] F. Rossetto and M. Zorzi, "On the design of practical asynchronous physical layer network coding," in *Signal Processing Advances in Wireless Communications, 2009. SPAWC '09. IEEE 10th Workshop on*, June 2009, pp. 469–473.

[66] P.-C. Wang, Y.-C. Huang, and K. Narayanan, "Asynchronous compute-and-forward/integer-forcing with quasi-cyclic codes," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, Dec. 2014, pp. 1504–1509.

[67] M. Biguesh and A. Gershman, "Training-based MIMO channel estimation: a study of estimator tradeoffs and optimal training signals," *Signal Processing, IEEE Transactions on*, vol. 54, no. 3, pp. 884 – 893, Mar. 2006.

[68] X. Guo and X. g. Xia, "A distributed space-time coding in asynchronous wireless relay networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 5, pp. 1812–1816, May 2008.

[69] F. Gao, R. Zhang, and Y.-C. Liang, "Optimal channel estimation and training design for two-way relay networks," *Communications, IEEE Transactions on*, vol. 57, no. 10, pp. 3024–3033, Oct. 2009.

[70] B. Jiang, F. Gao, X. Gao, and A. Nallanathan, "Channel estimation and training design for two-way relay networks with power allocation," *Wireless Communications, IEEE Transactions on*, vol. 9, no. 6, pp. 2022–2032, June 2010.

[71] L. Song, Y. Li, A. Huang, B. Jiao, and A. V. Vasilikos, "Differential modulation for bidirectional relaying with analog network coding," *Signal Processing, IEEE Transactions on*, vol. 58, no. 7, pp. 3933–3938, July 2010.

[72] K. Mizutani, Y. Kida, T. Miyamoto, K. Sakaguchi, and K. Araki, "Realization of TDD two-way multi-hop relay network with MIMO network coding," in *Proc. of the 6th Int. ICST Conf. on Cognitive Radio Oriented Wireless Networks and Commun.* Osaka, Japan: ICST, June 2011.

[73] D. Kramarev, Y. Hong, and E. Viterbo, "Software defined radio implementation of a two-way relay network with digital network coding," in *2014 Australian Commu-*

nications Theory Workshop (AusCTW), vol. 1, Sydney, Australia, Feb. 2014, pp. 120 – 125.

[74] L. Lu, L. You, Q. Yang, T. Wang, M. Zhang, S. Zhang, and S. C. Liew, "Real-time implementation of physical-layer network coding," in *Proceedings of the Second Workshop on Software Radio Implementation Forum*, ser. SRIF '13. New York, NY, USA: ACM, 2013, pp. 71–76.

[75] A. Marcum, J. Krogmeier, D. Love, and A. Sprintson, "Analysis and implementation of asynchronous physical layer network coding," *Wireless Communications, IEEE Transactions on*, vol. 14, no. 12, pp. 6595–6607, Dec. 2015.

[76] D. Kramarev, A. Sakzad, and E. Viterbo, "Implementation of a two-way relay network with compute-and-forward in GNU Radio," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 4, pp. 484–493, Apr. 2016.

[77] A. A. Abidi, "The path to the software-defined radio receiver," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 5, pp. 954–966, May 2007.

[78] J. Mitola, "The software radio architecture," *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26–38, May 1995.

[79] J. C. Richard Johnson, W. A. Sethares, and A. G. Klein, *Software Receiver Design*. Cambridge University Press, 2011.

[80] V. Alluri, J. Heath, and M. Lhamon, "A new multichannel, coherent amplitude modulated, time-division multiplexed, software-defined radio receiver architecture, and field-programmable-gate-array technology implementation," *Signal Processing, IEEE Transactions on*, vol. 58, no. 10, pp. 5369–5384, Oct. 2010.

[81] Y. M. Greshishchev, J. Aguirre, M. Besson, R. Gibbins, C. Falt, P. Flemke, N. Ben-Hamida, D. Pollex, P. Schvan, and S. C. Wang, "A 40GS/s 6b ADC in 65nm CMOS," in *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, Feb. 2010, pp. 390–391.

[82] S. Lee, A. Gerstlauer, and R. Heath, "Distributed real-time implementation of interference alignment with analog feedback," *Vehicular Technology, IEEE Transactions on*, vol. 64, no. 8, pp. 3513–3525, Aug. 2015.

[83] J. Vieira, S. Malkowsky, K. Nieman, Z. Miers, N. Kundargi, L. Liu, I. Wong, V. Owall, O. Edfors, and F. Tufvesson, "A flexible 100-antenna testbed for massive MIMO," in *Globecom Workshops (GC Wkshps), 2014*, Dec. 2014, pp. 287–293.

[84] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong, "Argos: Practical many-antenna base stations," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 53–64.

[85] J. Kerttula, N. Malm, K. Ruttik, R. Jäntti, and O. Tirkkonen, "Implementing TD-LTE as software defined radio in general purpose processor," in *Proceedings of the 2014 ACM Workshop on Software Radio Implementation Forum*, ser. SRIF '14. New York, NY, USA: ACM, 2014, pp. 61–68.

[86] M. Jain, J. I. Choi, T. Kim, D. Bharadia, S. Seth, K. Srinivasan, P. Levis, S. Katti, and P. Sinha, "Practical, real-time, full duplex wireless," in *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '11. New York, NY, USA: ACM, 2011, pp. 301–312.

[87] "Open BTS." [Online]. Available: http://openbts.org/

[88] Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner, "SODA: A low-power architecture for software radio," *SIGARCH Comput. Archit. News*, vol. 34, no. 2, pp. 89–101, May 2006.

[89] ——, "SODA: A high-performance DSP architecture for software-defined radio," *Micro, IEEE*, vol. 27, no. 1, pp. 114–123, Jan. 2007.

[90] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G. M. Voelker, "Sora: High-performance software radio using general-purpose multi-core processors," *Commun. ACM*, vol. 54, no. 1, pp. 99–107, Jan. 2011.

[91] A. Jow, C. Schurgers, and D. Palmer, "CalRadio: A portable, flexible 802.11 wireless research platform," in *Proceedings of the 1st International Workshop on System Evaluation for Mobile Platforms*, ser. MobiEval '07. New York, NY, USA: ACM, 2007, pp. 49–54.

[92] "REDHAWK Software Defined Radio Framework." [Online]. Available: https://redhawksdr.github.io/Documentation/

[93] "WARP Project." [Online]. Available: http://warpproject.org

[94] E. Grayver, *Implementing Software Defined Radio*. Springer New York, 2013.

[95] "GNU Radio." [Online]. Available: http://gnuradio.org/

[96] T. Rondeau, "GNU Radio scheduler details," GNU Radio, Tech. Rep., Sept. 2013. [Online]. Available: http://www.trondeau.com/blog/2013/9/15/explaining-the-gnu-radio-scheduler.html

[97] N. Truong, Y.-J. Suh, and C. Yu, "Latency analysis in GNU Radio/USRP-based software radio platforms," in *IEEE Military Communications Conference (MILCOM)*. San Diego, CA, USA: IEEE, Nov. 2013, pp. 305–310.

[98] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "An IEEE 802.11a/G/P OFDM receiver for GNU Radio," in *Proceedings of the Second Workshop on Software Radio Implementation Forum*, ser. SRIF '13. New York, NY, USA: ACM, 2013, pp. 9–16.

[99] T. W. Rondeau, T. O'Shea, and N. Goergen, "Inspecting gnu radio applications with controlport and performance counters," in *Proceedings of the Second Workshop on Software Radio Implementation Forum*, ser. SRIF '13. New York, NY, USA: ACM, 2013, pp. 65–70.

[100] T. Rondeau, N. McCarthy, and T. OShea, "SIMD programming in GNU Radio: Maintainable and user-friendly algorithm optimization with VOLK," in *Conference on Communications Technologies and Software Defined Radio (SDR'12)*. Brussel, Belgium: Wireless Innovation Forum Europe, June 2012. [Online]. Available: https://gnuradio.org/redmine/attachments/download/422/volk.pdf

[101] F. J. Harris, *Multirate Signal Processing for Communication Systems*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[102] "Reed-Solomon ECC Python extension module." [Online]. Available: https://pypi.python.org/pypi/reedsolomon

[103] L. Wei and W. Chen, "Efficient compute-and-forward network codes search for two-way relay channel," *Communications Letters, IEEE*, vol. 16, no. 8, pp. 1204–1207, Aug. 2012.

[104] A. Osmane and J.-C. Belfiore, "The compute-and-forward protocol: Implementation and practical aspects," July 2011, submitted to IEEE Communications Letters. [Online]. Available: http://arxiv.org/abs/1107.0300

[105] Y. H. Gan, C. Ling, and W. H. Mow, "Complex lattice reduction algorithm for low-complexity full-diversity MIMO detection," *Signal Processing, IEEE Transactions on*, vol. 57, no. 7, pp. 2701–2710, July 2009.

[106] A. Sakzad, E. Viterbo, Y. Hong, and J. Boutros, "On the ergodic rate for compute-and-forward," in *Network Coding (NetCod), 2012 International Symposium on*, June 2012, pp. 131–136.

[107] A. Sakzad, J. Harshan, and E. Viterbo, "Integer-forcing MIMO linear receivers based on lattice reduction," *Wireless Communications, IEEE Transactions on*, vol. 12, no. 10, pp. 4905–4915, Oct. 2013.

[108] A. S. J. Helberg and H. C. Ferreira, "On multiple insertion/deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 48, no. 1, pp. 305–308, Jan. 2002.

[109] P. Deutsch and J.-L. Gailly, "ZLIB compressed data format specification version 3.3," IETF RFC 1950, Tech. Rep., May 1996.

[110] T. C. Maxino and P. J. Koopman, "The effectiveness of checksums for embedded control networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 1, pp. 59–72, Jan. 2009.

[111] S. Heimlicher, M. Karaliopoulos, H. Levy, and M. May, "End-to-end vs. hop-by-hop transport under intermittent connectivity," in *Proceedings of the 1st international conference on Autonomic computing and communication systems*, no. 20. Rome, Italy: ICST, Oct. 2007.

[112] W. Fu, Z. Tao, J. Zhang, and D. Agrawal, "Error control strategies for WiMAX multi-hop relay networks," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, Nov. 2009, pp. 1–6.

[113] Z. Chen, C. Zhang, J. Zhang, and G. Wei, "ARQ protocols for two-way relay systems," in *6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*. Chengdu, China: IEEE, Sept. 2010, pp. 1–4.

[114] J. He and S. C. Liew, "ARQ for physical-layer network coding," *IEEE Transactions on Mobile Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[115] S. Wang, Q. Song, X. Wang, and A. Jamalipour, "Distributed MAC protocol supporting physical-layer network coding," *IEEE Transactions on Mobile Computing*, vol. 12, no. 5, pp. 1023–1036, May 2013.

[116] Ettus Research, "GPSDO Kit for USRP N200/N210," May 2014. [Online]. Available: https://www.ettus.com/content/files/gpsdo-kit_4.pdf

[117] ——, "Application note. synchronization and MIMO capability with USRP devices," Ettus Research, Tech. Rep., 2015. [Online]. Available: http://www.ettus.com/content/files/kb/mimo_and_sync_with_usrp_updated.pdf

[118] C. Sanderson, "Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments." NICTA, Australia, Tech. Rep., 2010. [Online]. Available: http://arma.sourceforge.net/

[119] U. Mengali and A. N. D'Andrea, *Synchronization Techniques for Digital Receivers*. New York: Plenum Press, 1997.

[120] H. Meyr, M. Moeneclaey, and S. A. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. New York: Wiley, 1998.

[121] G. Danesfahani and T. Jeans, "Optimisation of modified Mueller and Muller algorithm," *Electronics Letters*, vol. 31, no. 13, pp. 1032–1033, June 1995.

[122] F. M. Gardner, "A BPSK/QPSK timing-error detector for sampled receivers," *Communications, IEEE Transactions on*, vol. 34, no. 5, pp. 423–429, May 1986.

[123] J. R. Barry, A. Kavcic, S. W. LcLaughlin, A. Nayak, and W. Zeng, "Iterative timing recovery," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 89–102, Jan. 2004.

[124] C. Dick, F. Harris, and M. Rice, "Synchronization in software radios. carrier and timing recovery using FPGAs," in *Field-Programmable Custom Computing Machines, 2000 IEEE Symposium on*, 2000, pp. 195–204.

[125] F. J. Harris and M. Rice, "Multirate digital filters for symbol timing synchronization in software defined radios," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 12, pp. 2346–2357, Dec. 2001.

[126] X. Dang, Q. Li, and X. Yu, "Symbol timing estimation for physical-layer network coding," *IEEE Communications Letters*, vol. 19, no. 5, pp. 755–758, May 2015.

[127] Q. Yang, S. C. Liew, L. Lu, and Y. Shao, "Symbol misalignment estimation in asynchronous physical-layer network coding," *IEEE Transactions on Vehicular Technology*, accepted for publication.

[128] D. Kramarev, "Accurate symbol-level synchronization of universal software radio peripherals for physical-layer network coding applications," Oct. 2016, submitted to IEEE International Conference on Communications 2017.

[129] A. Eghbali, T. Saramaki, and H. Johansson, "On two-stage Nyquist pulse shaping filters," *Signal Processing, IEEE Transactions on*, vol. 60, no. 1, pp. 483–488, Jan. 2012.

[130] A. P. Vinod and E. M.-K. Lai, "Design of low complexity high-speed pulse-shaping IIR filters for mobile communication receiver," in *IEEE Int. Conf. on Circuits and Systems (ISCAS)*, IEEE. Kobe, Japan: IEEE, May 2005, pp. 352–355.

[131] V. Agarwal, P. Kim, D.-G. Oh, and D.-S. Ahn, "Hardware efficient root-raised-cosine pulse shaping filter for DVB-S2 receivers," in *Advances in Computing*

and Communications, ser. Communications in Computer and Information Science, A. Abraham, J. Lloret Mauri, J. Buford, J. Suzuki, and S. Thampi, Eds. Springer Berlin Heidelberg, 2011, vol. 191, pp. 595–603.

[132] N. Pal, R. Sarin, K. Singhal, and R. Rajan, "Distributed arithmetic algorithm for raised cosine filter in WCDMA system," *MIT International Journal of Electronics & Communication Engineering*, vol. 2, no. 1, pp. 5–10, Jan. 2012.

[133] Z. Wang, Y. Li, and H. Huang, "FPGA design and implementation of pulse shaping filter for coherent underwater communication," in *2012 2nd International Conference on Computer Science and Network Technology (ICCSNT)*, IEEE. Changchun, China: IEEE, Dec. 2012, pp. 748–752.

[134] A. Avizienis, "Signed-digit numbe representations for fast parallel arithmetic," *IRE Transactions on Electronic Computers*, vol. EC-10, pp. 389–400, Sept. 1961.

[135] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete powers-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 31, no. 3, pp. 583–591, June 1983.

[136] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 7, pp. 1044–1047, July 1989.

[137] Z. Tang, J. Zhang, and H. Min, "A high-speed, programmable, CSD coefficient FIR filter," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 4, pp. 834–837, Nov. 2002.

[138] J. Vankka, *Digital Synthesizers and Transmitters for Software Radio*. Springer US, 2005.

[139] V. Manoj and E. Elias, "Artificial bee colony algorithm for the design of multiplierless nonuniform filter bank transmultiplexer," *Information Sciences: an International Journal*, vol. 192, pp. 193–203, June 2012.

[140] J. Skaf and P. Boyd, "Filter design with low complexity coefficients," *Signal Processing, IEEE Transactions on*, vol. 56, no. 7, pp. 3162–3169, July 2008.

[141] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Upper Saddle River, NJ: Prentice-Hall, 1999.

[142] H. J. Oh, W. J. Oh, and Y. H. Lee, "Design of cascade-form IIR filters with powers-of-two coefficients using mixed integer linear programming," in *Circuits and Systems, 1996. ISCAS '96., Connecting the World., 1996 IEEE International Symposium on*, vol. 2, May 1996, pp. 221–224.

[143] L. Liang, M. Ahmadi, M. Sis-Ahmed, and K. Wallus, "Design of canonical signed digit IIR filters using genetic algorithm," in *The Thirty-Seventh Asilomar Conf. on Signals, Systems, and Computers*, vol. 2, IEEE. IEEE, Nov. 2003, pp. 2043–2047.

[144] T. Williams, M. Ahmadi, and W. Miller, "Design of 2D FIR and IIR digital filters with canonical signed digit coefficients using singular value decomposition and genetic algorithms," *Circuits, Systems and Signal Processing*, vol. 26, no. 1, pp. 69–89, Feb. 2007.

[145] B.-S. Chen, S.-C. Peng, and B.-W. Chiou, "IIR filter design via optimal Hankel-norm approximation," *Circuits, Devices and Systems, IEE Proceedings G*, vol. 139, no. 5, pp. 586–590, Oct. 1992.

[146] H. Brandenstein and R. Unbehauen, "Weighted least-squares approximation of FIR by IIR digital filters," *Signal Processing, IEEE Transactions on*, vol. 49, no. 3, pp. 558–568, Mar. 2001.

[147] "Ettus USRP X310." [Online]. Available: https://www.ettus.com/product/details/X310-KIT

[148] Y. Li, R. H. Louie, and B. Vucetic, "Relay selection with network coding in two-way relay channels," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 9, pp. 4489–4499, Nov. 2010.

[149] M. Zhou, Q. Cui, R. Jantti, and X. Tao, "Energy-efficient relay selection and power allocation for two-way relay channel with analog network coding," *Communications Letters, IEEE*, vol. 16, no. 6, pp. 816–819, June 2012.

[150] S. Talwar, Y. Jing, and S. ShahbazPanahi, "Joint relay selection and power allocation for two-way relay networks," *Signal Processing Letters, IEEE*, vol. 18, no. 2, pp. 91–94, Feb. 2011.