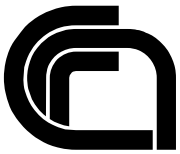


Consiglio Nazionale delle Ricerche

Assessing the Performance of a MIMO SDR Testbed with Dual Transceiver Implementation

V. Gardellin, A.Kocian, F. Martelli, P. Santi

IIT TR-02/2013

Technical report

Febbraio 2013

Istituto di Informatica e Telematica

Assessing the Performance of a MIMO SDR Testbed with Dual Transceiver Implementation

Vanessa Gardellin, Alexander Kocian, Francesca Martelli, Paolo Santi
Institute for Informatics and Telematics
Italian National Research Council
Via G. Moruzzi 1, 56124 Pisa, Italy
Email: {vanessa.gardellin,alexander.kocian,francesca.martelli,paolo.santi}@iit.cnr.it

***Abstract*—Software Defined Radio testbeds are becoming increasingly used in the wireless networking community, given their feature of leaving wireless network designer full control of the PHY layer. On the other hand, SDR testbeds are formed of very complex software/hardware tools, in which implementation bugs are likely and difficult to identify. For this reason, assessment of the results provided by an SDR platform should be a fundamental, preliminary step in the performance evaluation process.**

In this paper, we provide a thorough assessment of the MIMONet SDR platform for network-level exploitation of MIMO technology. To assess the platform, we have used two different implementations of an OFDM transceiver: one based on Matlab, the other on the GNU Radio software. We have then cross-validated performance by means of extensive measurements using the two alternative implementations. We have also designed and implemented a fine grained SNR and BER estimation methodology, that allowed us to carefully validate performance of the two software implementations against theoretical predictions. When collectively considered, the results of our measurements promote MIMONet as the first SDR testbed with carefully validated performance.

***Index Terms*—Software Defined Radios; MIMO testbed; OFDM; performance assessment.**

I. Introduction

The increasing use of wireless technologies to support communication needs of individuals, companies, organizations, communities, etc., and the consequent increasing wireless traffic demand, will pose the problem of efficient bandwidth utilization at the forefront. Advanced wireless communication techniques such as cognitive radios [1] and multi-antenna systems [2] are then expected to be increasingly used by wireless network designers to improve bandwidth utilization.

A challenge to face when designing network-level approaches that make use of advanced communication technologies is their evaluation in a real environment. In fact, cross-layer design (down to the PHY layer) is often the key to fully exploit the potential of these technologies at network-level. Thus, the wireless network designer should be able to control and modify not only the upper layers of the network architecture, but also the PHY layer. Typically, commodity wireless networking cards do not allow access to the PHY layer, since the low level functionalities realized at the PHY layer such as DSP are implemented in hardware. Thus, specially designed hardware/software platforms such as software defined radios [3] must be used.

In software defined radios (SDRs), the entire transmitter and receiver chains are implemented in software, while a specialized hardware connected to the software modules is used as the radio front-end. This way, the wireless network designer gains full control of the PHY layer functionalities, enabling implementation and testing of the designed cross-layer, network-level protocols.

While SDR platforms give the designer the power of fully controlling PHY layer design, they also expose the designer to the risk of making programming errors. In fact, SDRs are very complex hardware/software tools, often based on open source frameworks developed by different programmers. In such a situation, the risk of having bugs in the implementation is very high, possibly leading to inaccurate results when, say, a networking protocol is tested. For this reason, a SDR testbed should be carefully assessed before it can be used for further testing. To our best knowledge, however, the problem of assessing performance of an SDR platform has not been considered in the literature so far.

In this paper, we present a thorough performance assessment of the MIMONet SDR platform for testing network-level MIMO protocols we are currently developing. MIMONet performance assessment is achieved by means of (i) *cross-validation* and (ii) *comparison of SISO vs. SIMO*. As for (i), MIMONet features two alternative software implementations of a SISO OFDM system: the first implementation is based on the open source GNU Radio framework [4], while the second implementation is based on Matlab. Performance of the two alternative implementations have been cross-validated by building BER vs. SNR for the two approaches, and by comparing them with theoretical predictions. As for (ii), we have tested the performance of a SISO system and of different SIMO systems, and compared the gain provided by SIMO vs. SISO with theoretical predictions.

The specific technical contributions of this paper are:

– the realization of the first SDR testbed featuring two alternative, fully fledged OFDM implementations, which allow cross-validation of the obtained results;
– the implementation of a set of tracing tools for performance evaluation, that allow estimating SNR values observed on the single sub-carriers and building channel profiles in both the time and the frequency domain;
– the design of a systematic assessment methodology, based

on building theoretical as well as measured BER vs. SNR and PER vs. SNR curves. The BER curves are used to define a notion of *effective SNR* [5] aimed at accounting for a channel profile in the SNR definition.

We have verified that the performance provided by the MIMONet testbed is in full agreement with theoretical predictions. Interestingly, our study has revealed an unexpected inefficiency of the GNU Radio implementation, which systematically experiences lower effective SNR values than the corresponding Matlab implementation. Our tracing tools allowed us to identify the cause of such inefficiency in the fact that in GNU Radio central sub-carriers systematically experience much lower SNR values that non-central sub-carriers, highlighting that current GNU Radio implementation only partially compensates the direct current (DC) offset generated by the Universal Software Radio Peripheral (USRP) units used as radio front-end.

II. Related Work

A number of SDR-based testbeds have been recently introduced in the literature to test network-level protocols. In terms of hardware, the most used SDR platforms are USRP produced by Ettus Research^{TM} [6], WARP [7] developed at Rice University, and Microsoft Sora [8]. One of the first SDR platforms for network MIMO systems is Hydra [9] developed at UT Austin, which is based on USRP hardware and GNU Radio software. Hydra provides a PHY layer implementation based on IEEE 802.11n, and a MAC layer based on the Distributed Coordination Function of IEEE 802.11. The Hydra platform has been used in [10] to evaluate performance of a rate adaptation technique for multi-antenna systems.

USRP hardware coupled with GNU Radio software is also used in a series of papers by Katabi et al. [11], [12], [13], where the performance of interference alignment and cancellation [11], of a random access protocol for MIMO networks [12], and of a rate adaptation technique for MIMO networks [13] are tested. WARP-based platforms have instead been used to test performance of beamforming in [14] and [15]. Finally, a Sora-based platform has been used in [16] to test performance of a spatial multiple access protocol for wireless LANs.

Similarly to Hydra and to the testbeds used in [11], [12], [13], the MIMONet platform is based on USRP hardware, and uses GNU Radio as software platform. However, a distinguishing feature of MIMONet is that we have also implemented an alternative transceiver design based on Matlab, mirroring as much as possible the transmitter/receiver chains implemented in GNU Radio. To our best knowledge, MIMONet is the first SDR testbed featuring such a dual transceiver implementation, whose performance is carefully assessed by means of cross-validation and comparison with theoretical predictions.

III. MIMONet Architecture

In this section we describe the MIMONet testbed architecture, starting with a description of the hardware platform in Section III-A, and then presenting the two alternative software transceiver implementations in sections III-B and III-C.

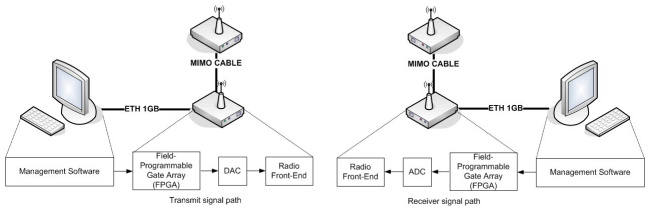


Figure 1. A transmitter and a receiver node.

A. Hardware

A transmitter and a receiver nodes of the MIMONet testbed are illustrated in Fig. 1. Each node mainly includes: (i) a PC equipped with an Intel®Core^{TM} i7-2600 at 3.4GHz and 8 GB of main memory; (ii) up to two USRPs model N210 by Ettus Research^{TM} [6]; (iii) a MIMO cable by Ettus Research^{TM} [6]; and (iv) two Gigabit Ethernet cables.

USRPs are a family of computer-hosted hardware designed *with* GNU Radio. A single USRP unit consists of a motherboard and a daughterboard, where the latter is the radio front-end. The USRP N210 was chosen as the motherboard and the XCVR2450 as the daughterboard due to their computational capabilities, as well as their ability to operate in the frequency ranges of $[2.4; 2.5]$ GHz and $[4.9; 5.85]$ GHz. In fact, those are the frequencies used by the IEEE 802.11n standard for multiple antenna wireless networks [17]. When a USRP is used as a transmitter, the management software on the PC produces a signal that is converted from digital to analog (DAC) so that it can be suitable for the radio front-end for subsequent transmission over the air. On the receiver side, the radio front-end receives the signal which is converted from analog to digital (ADC) and sent to the management software in the PC. A USRP has an FPGA in it where it is implemented a DC offset removal algorithm. This algorithm is not perfect and the degree of imperfection changes with center frequency. The DC offset is usually very narrowband in nature, and for many types of modulations has negligible effects. Moreover, a local oscillator (LO) leakage produced by the analog mixer contributes to a small DC offset appearing in the signals coming out of that mixer. For the reasons above, recommendations given by Ettus Reasearch^{TM} suggest to leave the two central sub-carriers in an OFDM system unused.

A Gigabit Ethernet cable is used to connect PCs and USRPs, while the connection between the two USRPs (in case of multi-antenna configuration) is provided by a so-called MIMO cable. The MIMO cable is designed to guarantee synchronization between the two units, i.e., coherence in sampling clocks and local oscillators, which is essential for implementation of any MIMO technique. A MIMO cable can be used in shared or dual modes. We choose the *shared Ethernet mode* where one USRP, called master, is connected to the Ethernet cable and the other USRP, called slave, receive clock reference, time reference and *data* from the master over the MIMO cable. This method differs from the dual Ethernet mode, where both USRPs are connected to the PC through an Ethernet cable,

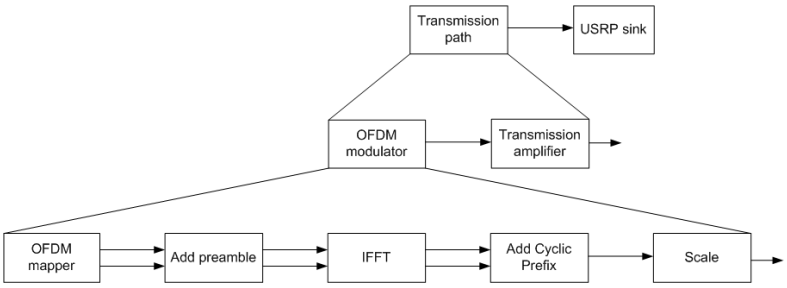


Figure 2. GNU Radio transmit chain

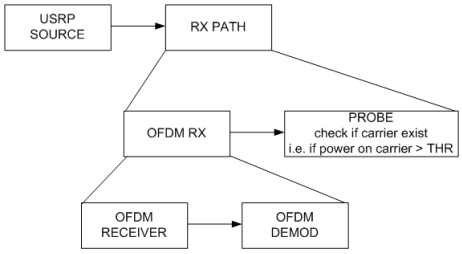


Figure 3. GNU Radio receiver chain

and no data are communicated over the MIMO cable. The rationale behind our choice is an easier implementation and synchronization of data to transmit.

B. GNU Radio Implementation

GNU Radio is an open source software framework developed to implement Software Defined Radios (SDR). Given its scalability, its flexibility in setting the signal processing components, and its wide user base, GNU Radio is adopted as one of the two software platforms for transceiver implementation in MIMONet. Functionalities already implemented in GNU Radio, and that we use as given, are: (i) the Orthogonal Frequency Division Multiplexing (OFDM) mechanism where a transmitted/received signal is split in a subset of independently modulated signals distributed on orthogonal subcarriers; (ii) synchronization and packet detection; (iii) several modulation schemes, among which the binary phase-shift keying (BPSK), quadrature PSK (QPSK) and 8PSK considered in this study.

We have added and enhanced several features of GNU Radio in order to have an SDR framework as close as possible to an IEEE 802.11n node [17], the major contribution being the *forward error correction* (FEC) module. The used FEC is based on convolutional encoding and Viterbi decoding as suggested in IEEE 802.11n standard [17]. The encoding is an industry-standard generator polynomials ($g0 = 133$ and $g1 = 171$) of rate $R = 1/2$. FEC is computed on the packet payload and on its cyclic redundancy check (CRC), namely the entire packet except the header; the header cannot be encoded because the information contained in it is used to reconstruct the encoded packet.

In the following, we described the GNU Radio software architecture highlighting the transmitter and receiver chains. The main blocks of the transmission chain are shown in Fig. 2: (i) an application that generates packets, (ii) an OFDM mapper that produces as many OFDM symbols as needed in order to hold a full packet according to the modulation scheme used, (iii) a preamble adder that inserts pre-modulated known symbol before each packet in order to perform synchronization and to estimate the communication channel at the receiver, (iv) a pilot adder that occupies 30 subcarriers in each OFDM symbol (pilots are used at the receiver to correct phase rotation), (v) an Inverse Fast Fourier Transform (IFFT) to convert symbols from frequency domain to time domain, (vi) a cyclic prefix adder that copies a portion of symbols located at the end of the OFDM symbol, at the beginning of the same symbol, and finally, (vii) a scale factor that normalizes the signal power of symbols and it is required by the hardware.

Figure 3 shows the blocks that build the receiver chain. The main block is the OFDM demodulator which performs synchronization on the received data, demodulates symbols into bits and packs bits into packets. The synchronization stage is the most complex because it performs: (i) frame detection, (ii) Schmidl and Cox preamble correlation [18], (iii) cyclic prefix removal, (iv) Fast Fourier Transformation (FFT), and, (v) sub-carriers equalization.

Schmidl and Cox preamble correlation is performed to obtain time synchronization and fine frequency offset correction (see next section). Sub-carrier equalization is performed using the least square estimation technique based on the received preamble and the known symbols in order to remove the distortion introduced by the communication channel.

C. Matlab

To benchmark the GNU-based transceiver and cross-validate performance, a second software platform has been used. For the reasons detailed below, we chose the platform MATLAB®/Simulink®:

+ Matlab is optimized for vector/matrix based operations (loops are time-consuming);
+ Matlab is very good at generating plots;
+ knowledge on how to program Matlab is available at major institutions;
+ professional technical support;
+ large community of users that share numerical codes;
+ Simulink provides a graphical wrapper of Matlab.

On the other hand, we are aware of Matlab limitations and cons, including:

- license fee;
- processing speed comparable with other programming languages (e.g., C++ used in GNU Radio) only if the Parallel-computing feature is used;
- do not support the USRP MIMO cable; synchronization with external clock required for realizing multi-antenna systems.

The latter limitation in particular constrained our current Matlab implementation to the SISO configuration.

The Matlab/Simulink-based transmitter and receiver are depicted in Fig. 4 and Fig. 5, respectively.

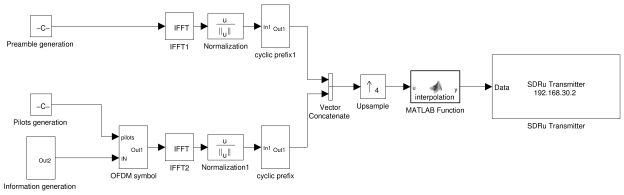


Figure 4. Matlab transmitter chain.

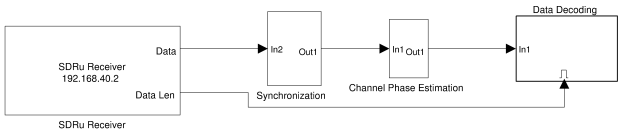


Figure 5. Matlab receiver chain.

In the source-coder, the i.i.d. information bit sequence $\boldsymbol{b} \in \{0,1\}^{N_b}$ is encoded at rate R, and mapped into the data sequence $\boldsymbol{d} \in \{-1,+1\}^{N_d}$, $N_d = N_b/R$, $N_b/R \mod N_d = 0$. The code vector $\boldsymbol{d}$ is interleaved (pseudo-randomly) and partitioned into OFDM symbols to which $N_p = 30$ BPSK modulated pilot symbols are added. The resulting data vectors are processed by the IFFT matrix $F^{-1} \in \mathbb{C}^{N\times N}$, $N \mod N_d = 0$ and fed into the permutation matrix Π_{t}, to add a cyclic prefix of length N_{cp}. A packet is formed of one preamble (upper branch in Fig.4) and L OFDM symbols (lower branch in Fig. 4). The preamble, which is essential for the receiver to synchronize on the data stream in a coarse fashion, consists of two equal halves in the time domain i.e., every second tone is occupied only in the frequency domain. To improve SNR at the receiver, the resulting signal is upsampled by a factor $Q = 4$ and sent to the transmitter.

In the receiving USRP, the signal is downconverted, lowpass-filtered, and sampled. The resulting sequence $\{\boldsymbol{r}\} \in \mathbb{C}$ is forwarded to the Matlab/Simulink platform.

1) Acquisition: The first module in the receiver chain, provides coarse synchronization (acquisition). Since the receiver does not know time offset or frequency offset, estimation is required. Assume the channel is non-dispersive. Then, the time offset can be modeled as a (single) delay in the channel response $\delta(m-\theta)$ with the integer arrival time θ. Moreover, the local oscillators in the transmitter and receiver cause a frequency offset ε of the subcarriers. This frequency offset can be modeled as multiplicative distortion in the time domain $e^{j2\pi\varepsilon m/N}$. The received signal then reads

$$r_m = d(m-\theta)e^{j2\pi\varepsilon m/N} + n(m)\ ,$$

where $n(m)$ is complex additive white Gaussian noise.

As the preamble contains two equal halves, its cross-correlation at delay index τ is given by [18]

$$\mathcal{P}(\tau) = \sum_{m=0}^{N/2-1} r^*_{\tau+m} r_{\tau+m+N/2};$$

and its energy can be computed as

$$\mathcal{R}(\tau) = \sum_{m=0}^{N/2-1} |r_{\tau+m+N/2}|^2.$$

The resulting (normalized) timing metric

$$\mathcal{S}(\tau) = \frac{|\mathcal{P}(\tau)|^2}{\mathcal{R}(\tau)^2}$$

has a plateau, centered around timing offset $\tau = 0$. A plateau is considered as successfully detected if $\mathcal{S}(\tau)$ is larger than a predefined threshold $\eta = 0.8$.

2) Tracking: We want to find an estimate jointly for θ and ε, given the observation $\boldsymbol{r}$ in the time domain. Under the assumption that $\boldsymbol{r}$ is a jointly Gaussian vector, it can be shown [19] that the likelihood function for (θ,ε) is given by

$$\log p(\boldsymbol{r}|\theta,\varepsilon) = |\gamma(\theta)|\cos(2\pi\varepsilon + \angle\gamma(\theta)) - \Phi(\theta) \qquad (1)$$

Here, $\angle$ denotes the argument of the complex number. Moreover,

$$\gamma(m) \triangleq \sum_{k=m}^{m+N_{cp}-1} r(k)r^*(k+N)$$

where N_{cp} is the length of the cyclic prefix and,

$$\Phi \triangleq \frac{1}{2}\sum_{k=m}^{m+N_{cp}-1} |r(k)|^2 + |r(k+N)|^2\ .$$

We want to maximize the likelihood function in (1) with respect to ε and θ, yielding

$$\hat{\theta}_{ML} = \arg\max_{\theta}\{|\gamma(\theta)| - \Phi(\theta)\}$$

and

$$\hat{\varepsilon}_{ML}(\theta) = -\frac{1}{2\pi}\angle\gamma(\theta) + I.$$

Notice that the integer frequency offset I cannot be resolved at this step.

3) Integer Offset and Phase Offset: Assuming perfect time-offset compensation and frequency-offset compensation up to the integer part I, we can use the pilot symbols in each OFDM interval to retrieve integer offset and phase offset [20]. Maximizing the likelihood function of $\boldsymbol{r}$ given I and phase-offset ϕ in the frequency domain, leaves

$$\hat{I} = \arg\max_{I} |\boldsymbol{d}_{N_p}[I]^H \boldsymbol{R}|, \hat{\phi} = \angle(\boldsymbol{d}_{N_p}[\hat{I}]\boldsymbol{R}).$$

Here, $\boldsymbol{R}$ denotes the Fourier transform of r.

Having synchronized, the resulting signal is hard demodulated and fed into a Viterbi decoder to reconstruct the data bits.

IV. BER Analysis

In this section, we provide an analytical expression for the bit-error-performance of OFDM over real communication channels.

In the communication system, N_b source bits are encoded and (randomly) interleaved into N_d code bits, i.e. rate $R = N_b/N_d$, modulated by symbols from a modulation alphabet of size M. The random interleaver prevents burst errors. This information bit-stream is transmitted over a packet containing L OFDM symbols, each having N sub-carriers. Hence, one packet carries $LN\log_2(M)$ code bits and $LN\log_2(M)N_b/N_d$ information bits, respectively. Perfect synchronization is assumed throughout the derivations.

A. Single-Carrier Modulation

To get started, let us assume the number of sub-carriers in each OFDM symbol is equal one. Then, the bit-error probability of maximum-likelihood sequence decoder, P_ε can be upper bounded by [21]

$$P_\varepsilon \leq \int_{\gamma_b=-\infty}^{+\infty} \sum_{d=d_{\text{free}}}^{\infty} \beta_d P_2(d,\gamma_b) p(\gamma_b)\ d\gamma_b. \tag{2}$$

where $P_2(d,\gamma_b)$ is the pairwise-error-probability, i.e., the probability that the decoded sequence at the receiver differs from the transmitted one by Hamming distance of d bits at a given per-bit SNR γ_b; β_d indicates the number of information bit errors in selecting an incorrect path that merges with the all-zero path at some node in the trellis; and d_{free} is the free distance of the code. Finally, $p(\gamma_b)$ denotes the probability density function of the observed SNR, which has been obtained from measurements in our case. For hard-decision decoding, the pairwise-error-probability is [22]

$$P_2(d,\gamma_b) = \sum_{r=(d+1)/2}^{d} \binom{d}{r} P_b(\gamma_b)^r \left(1-P_b(\gamma_b)\right)^{(d-r)}, \tag{3a}$$

$$\begin{aligned} P_2(d,\gamma_b) = &\sum_{r=d/2+1}^{d} \binom{d}{r} P_b(\gamma_b)^r \left(1-P_b(\gamma_b)\right)^{(d-r)} \\ &+\frac{1}{2}\binom{d}{d/2} P_b(\gamma_b)^{(d/2)} \left(1-P_b(\gamma_b)\right)^{(d/2)} \end{aligned} \tag{3b}$$

for odd and even values of d, respectively, where $P_b(\gamma_b)$ denotes the probability of a bit error for the binary symmetric channel with a SNR of γ_b.

Under the assumption of BPSK modulation, bit error probabilities can be approximated as follows [23]:

$$P_b(\gamma_b) \approx Q\left(\sqrt{2\gamma_b \frac{k}{n}}\right)$$

where $Q(\cdot)$ is the tail probability of the standard normal distribution. For 8-PSK modulation, we have [23]:

$$P_b(\gamma_b) \approx \frac{2}{3} Q\left(\sqrt{3\gamma_b \frac{k}{n}\left(1-cos\frac{\pi}{4}\right)}\right).$$

B. Multi-Carrier Modulation

The bit-error analysis for single-carrier modulation is valid also for the multi-carrier setting, as long as the memory of the communication channel is less than the length of the cyclic prefix. Under this assumption, the cyclic prefix prevents inter-symbol interference. Moreover, when the channel is quasi-static, i.e., the Doppler shift is negligible w.r.t. to the sub-carrier spacing, there is negligible inter-carrier interference. Under these assumptions, each sub-carrier fades independently, and the average bit-error-probability yields

$$\overline{P_\varepsilon} = \frac{1}{N}\sum_{i=1}^{N} P_\varepsilon[i]. \tag{4}$$

Here, $P_\varepsilon[i]$ denotes the bit-error-probability for carrier i, derived in Eq. (2).

V. Sub-carrier SNR estimation

A. SNR computations

In our OFDM setting, the FFT size is 512, the number of used sub-carriers (namely, occupied tones) is 200 and the cyclic prefix $N_{cp} = 128$. As suggested by GNU Radio team and Ettus support, the two central sub-carriers are left empty, so actually the number of sub-carriers used is $N = 198$. For each sub-carrier i, we compute the average (symbol) SNR, denoted $\gamma_{s,i}$, as follows. Since SNR is evaluated by means of noise variance estimation (see below), we have first computed by numerical evaluation the minimum number of samples $\#_{min}$ of a normally distributed random variable $\mathcal{N}$ (noise) which must be observed to obtain an accurate estimate of the variance of $\mathcal{N}$. Clearly, $\#_{min}$ depends on the variance of the observed variable $\mathcal{N}$, with higher $\#_{min}$ values needed for higher variances of $\mathcal{N}$. Considering that we expect to operate our system in a regime where an SNR of at least $0dB$ is achieved, we have made the worst-case assumption of a variance for $\mathcal{N}$ resulting in an SNR of $0dB$. Under this assumption, $\#_{min}$ turned out to be around 100 samples. Since in our setting one sample corresponds to one OFDM symbol, we have defined a number of OFDM symbols used for estimating SNR values close to 100. For instance, with BPSK modulation we estimate SNR every 11 packets, since every packet is composed of 9 OFDM symbols, and we then use 99 samples to estimate SNR.

In Table I we report the number of OFDM symbols per packet when packet payload is equal to 100 bytes, the number of packets used for computing one SNR value, and the total numbers of packets transmitted in one experiment run, with respect to the modulation.

Modulation	OFDM symbols (L)	Packets to compute SNR	Transmitted packets
BPSK	9	11	10500
QPSK	5	25	21000
8PSK	3	33	35000

Table I
Number of OFDM symbols per packet, number of packets used to compute SNR, and total number of transmitted packets for different modulations

By definition, in a channel i, the SNR is given by

$$\gamma_i = \frac{\epsilon_s |h_i|^2}{N_0} = \frac{1}{\mathrm{var}(\mathbf{n})}$$

The symbol SNR is computed by estimating the noise variance around the constellation points and normalized to the received signal power, namely

$$\gamma_s = \frac{1}{\mathrm{var}(\mathbf{n})R}$$

where R is the coding rate (in our case 1/2). The SNR per bit is given by

$$\gamma_b = \frac{\gamma_s}{\log_2(M)}$$

where M is the modulation used.

To build the profile of our channel, SNR values are divided into bins: each bin is $0.5dB$ wide and the range considered is [-2,30] (i.e 65 bins, $B = 65$ in the following), where all values less than $-2dB$ (greater than $30dB$) are grouped in the first (last) bin.

For each sub-carrier, every time a new SNR value is estimated, it is put in the proper bin. At the end of each experiment run, we count the occurrences in each bin for all sub-carriers, and their probabilities are computed. For each subcarrier i and for each bin j, we compute the probability of occurrences p_{i,γ_j} as

$$p_{i,\gamma_j} = \frac{\#(\gamma \ in \ bin_{i,j})}{\#(\gamma \ per \ subcarrier \ i)}$$

Then, we compute also the average SNR value observed in each sub-carrier as

$$\gamma_i = \sum_{j=1}^{B} \gamma_i p_{i,\gamma_j}.$$

B. Effective SNR

The notion of *effective SNR* has been introduced in [5] and also considered in [24]. It is defined as the SNR value that would give the same bit error performance on a narrow-band channel. In other words, the SNR obtained by simply averaging the SNR values of all sub-carriers (for instance, computed on a per-packet basis) is not representative of the actual goodness of the channel because it does not take into account the possible occurrence of weaker carriers caused by frequency selective fading. Ideally, the measured average SNR value matches the effective SNR only in case of a perfectly flat channel.

To compute the effective SNR, we need to use an analytical function f that, given an SNR value, returns the expected BER. By reversing this function, it is then possible to compute the effective SNR (SNR_{eff}) given the average SNR (SNR_{avg}) as $SNR_{eff} = f^{-1}(f(SNR_{avg}))$.

In [24], the authors use as function f simply the *uncoded upper bound* which holds only in case of AWGN channel with uncoded transmissions, i.e., the standard normal CDF (in case of BPSK modulation we have $f(\gamma_s) = Q(\sqrt{2\gamma_s})$, where γ_s is the symbol SNR).

We have modified such a definition to take into account both encoding and the channel effect, as described in the following. First, at the end of each experiment run (namely, for each channel realization), we take all SNR values, and for each carrier n we compute γ_i as described in the previous section.

Then, each bin $\gamma_{i,j}$ is associated with a theoretical BER estimate, $P_{\epsilon_{\gamma_{i,j}}}$, obtained from the coded approximation BER_{coded} computed as detailed in Section IV:

$$P_{\epsilon_{\gamma_{i,j}}} = BER_{coded}(\gamma_{i,j}) \ .$$

Finally, these BER values are weighted with their probability of occurrence, and we get an approximated BER value for subcarrier i:

$$P_{\epsilon_{appr,i}} = \sum_{j=1}^{B} P_{\epsilon_{\gamma_{i,j}}} p_{\gamma_{i,j}} \ .$$

We repeat this procedure for all sub-carriers, and finally we obtain an approximated BER value for the specific channel profile resulting from our measurements:

$$P_{\epsilon_{appr}} = \frac{1}{N} \sum_{i=1}^{N} P_{\epsilon_{appr,i}} \ . \tag{5}$$

The BER value computed in equation (5) is then coupled with the average SNR observed across all sub-carriers in the experiment run, which is given by

$$\gamma_{avg} = \frac{1}{N} \sum_{i=1}^{N} \gamma_i \ .$$

So, for each channel realization (experiment run), we generate a pair $(\gamma_{avg}, P_{\epsilon_{appr}})$. By repeating experiment runs with different transmitter and receiver parameters of the USRPs for the same channel (positions of transmitter and receiver unchanged), we can thus generate an approximation of the performance that could be reached with the specific channel at hand. We call such a curve *effective BER*, $P_{\epsilon_{eff}}$. More specifically, since $P_{\epsilon_{eff}}$ is formed of a discrete set of points, we considered a fitting curve $P'_{\epsilon_{eff}}$ as the function f needed to compute the effective SNR. We can then use the reverse function f^{-1} to compute the effective SNR value corresponding to any observed average SNR value.

VI. Performance assessment: Matlab vs. GNU

In this section, we compare Matlab and GNU Radio implementations in terms of SNR performance achieved. In our measurement campaign, we considered two different scenarios: *line-of-sight* (LOS) and *non-line-of-sight* (NLOS), both in an indoor office environment. In the LOS scenario, the transmitter and receiver nodes are located in the same room, 3 meters apart and one in front of the other; in the NLOS scenario, the two nodes are located in two different rooms, 7 meters apart, and between them there are walls of bricks, steel and wood. Transmitter and receiver parameters of the USRPs are varied to obtain measures with different SNR values.

A. Setup

In both implementations, the parameters at the transmiiter and receiver were set as specified in Table II.

Parameter	value@TX	value@RX
Center frequency (Hz)	$2.415e9$	$2.415e9$
Local Oscillator offset (Hz)	0	0
Gain (dB)	$< 0..35 >$	$< 0..80 >$
Sample Rate (Hz)	$200e3$	$200e3$

Table II
USRP SETTINGS AT THE TX AND RX SITE

Other parameters are: (i) payload packet size is set to 100 (GNURadio and Matlab), 500 (GNU Radio) or 1000 (GNU Radio) bytes; (ii) coding rate $R = 1/2$ (iii) generator polynomial $G = (171, 133)_8$ (according to IEEE 802.11n standard); (iv) FFT size 512 and cyclic-prefix length 128; (v) BPSK, QPSK and 8PSK modulation.

B. Comparison

In Figures 6 and 7 we show the semi-analytical BER estimation curve obtained taking into consideration the specific profiles of our LOS and NLOS channels, as described in Section V-B. The figures also report the BER vs. SNR curves obtained with GNU Radio and Matlab implementations. Notice that the floor effect in the measured curves is due to the fact that, with the number of transmitted packets in a single experiment run, we cannot reliably measure BER values below 10^{-3}. As it is evident from the figure, the Matlab implementation achieves better BER values for same SNR as compared to GNU Radio. This is most likely due to the fact that with GNU Radio implementation central sub-carriers systematically experience relatively lower SNR values. This effect can be seen in Figure 8, reporting the SNR profile in the frequency domain for a specific channel realization: the central sub-carrier experiences a close to $5dB$ lower value than average SNR. As a result of this, the effective SNR of that specific channel realization is $8.4dB$, more than $1dB$ lower than the average SNR of the profile. On the other hand, with Matlab we do not have such effect, as shown in Figure 9. However, Matlab implementation suffers a slope in the SNR profile, likely caused by the fact that pilots for phase correction are grouped at the beginning of the occupied tones. This effect is however less detrimental than the considerable SNR drop for central sub-carriers experienced by GNU Radio, as witnessed by the relatively higher effective SNR value of the Matlab implementation ($10.4dB$ instead of $8.4dB$) for a comparable average SNR value of the profile. The different behavior of Matlab and GNU Radio implementations in the frequency domain is well appreciable in the three-dimensional plots reported in Figures 10 and 11.

VII. PERFORMANCE ASSESSMENT: SISO VS. SIMO

In this section we present the results of the comparison between SISO and SIMO performance with GNU Radio implementation. As explained in Section III-C, Matlab does not support the MIMO cable used to synchronize two USRP

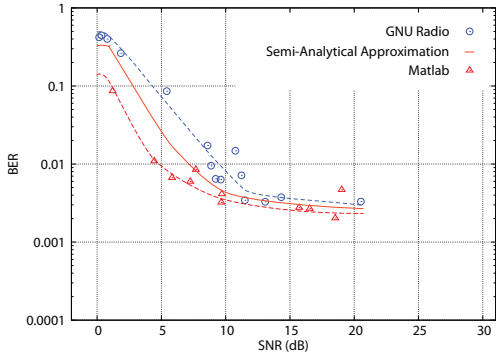


Figure 6. Semi-analytical BER approximation in LOS configuration and measured BERs for Matlab and GNU Radio implementations.

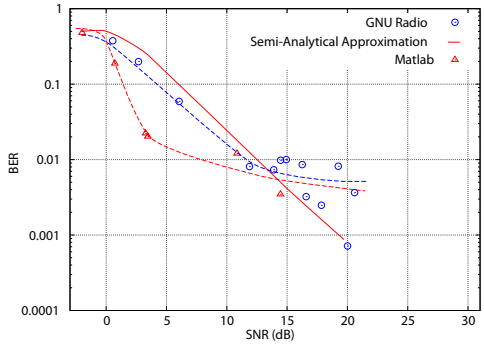


Figure 7. Semi-analytical BER approximation in NLOS configuration and measured BERs for Matlab and GNU Radio implementations.

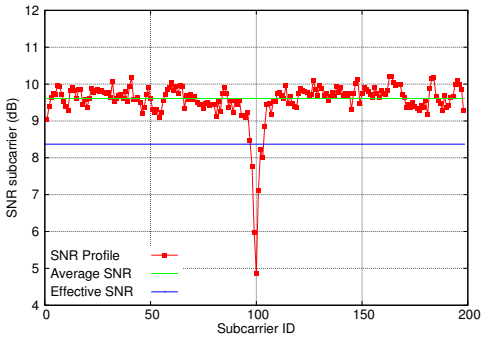


Figure 8. GNU Radio SNR profile in a LOS channel realization.

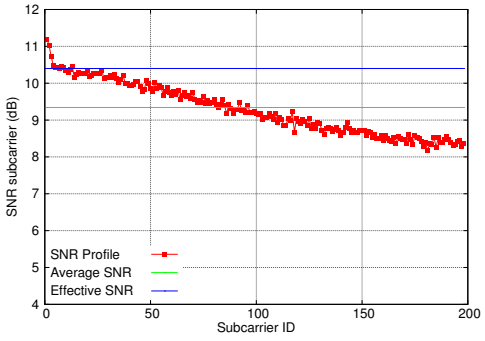
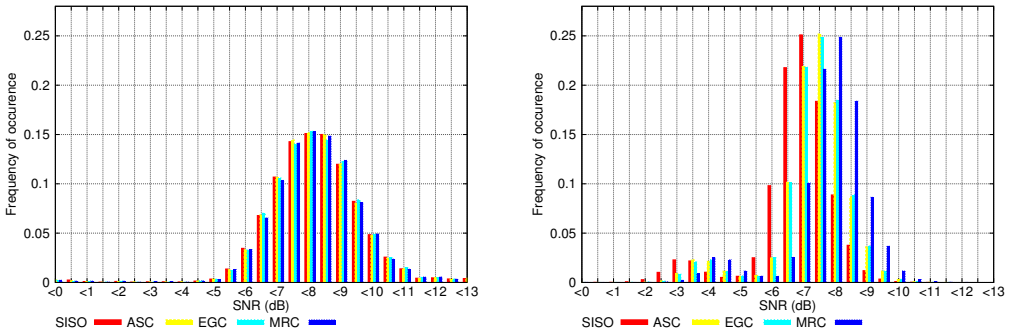


Figure 9. Matlab SNR profile in a LOS channel realization.



(a) BPSK LOS

(b) 8PSK LOS

(c) BPSK NLOS

(d) 8PSK NLOS

Figure 12. LOS vs. NLOS

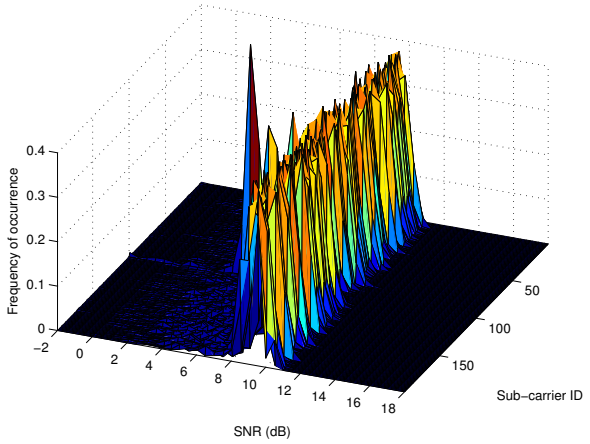


Figure 10. GNU Radio 3D profile in the channel realization corresponding to the SNR profile in Figure 8.

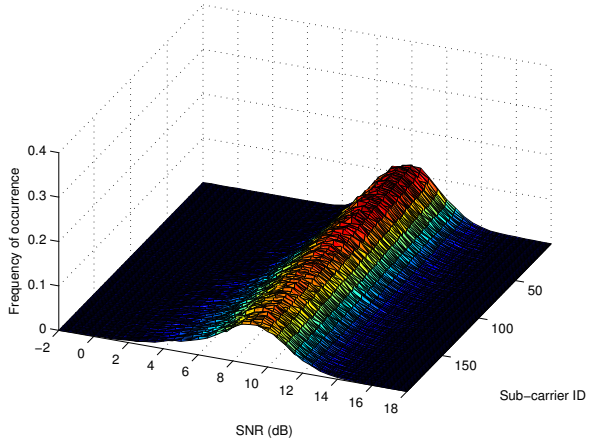


Figure 11. Matlab 3D profile in the channel realization corresponding to the SNR profile in Figure 9.

units, so we could not perform experiments with Matlab in SIMO configuration.

Figure 12 shows the comparison of the SNR probability mass function observed in the LOS and NLOS scenarios with two modulations (BPSK and 8PSK), in both SISO and SIMO configuration. The SIMO configuration uses antenna selection (ASC), equal gain combining (EGC), or maximum ratio combining (MRC) at the receiver. Transmitter and receiver gains are chosen is such a way that the resulting average SNR is about $9dB$ in both LOS and NLOS configuration.

It is interesting to observe that the SNR profile is more concentrated around the average value in the LOS scenario with both modulation schemes, as compared to the NLOS scenario. This is in accordance with expectations, given the relatively more considerable multipath effect in NLOS conditions. Furthermore, it is interesting to observe that the beneficial effect of receiver diversity techniques becomes evident in NLOS conditions. Again, this is in accordance with expectations, since receiver diversity techniques are expected to yield negligible benefit when the two antennas observe highly correlated channels (LOS conditions), and a noticeable benefit when the two antennas observe uncorrelated channels (NLOS conditions). It is also interesting to observe that the beneficial effect of receiver diversity techniques is more evident for higher order modulation schemes, again in accordance with theoretical predictions. Finally, MRC provides the best perfor-

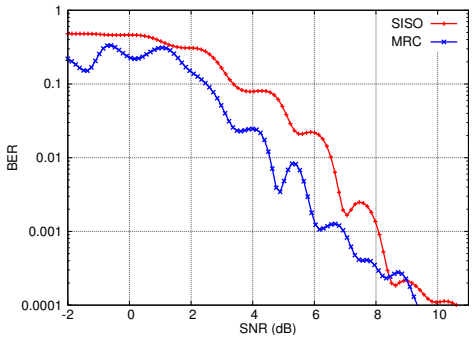


Figure 13. SISO vs. MRC in terms of bit error rate (BER).

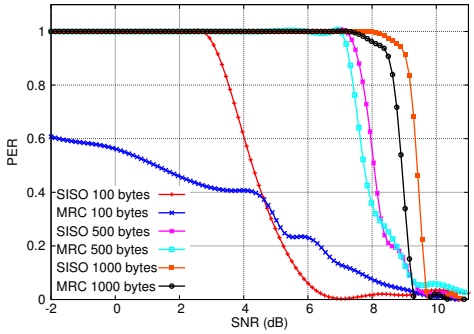


Figure 14. SISO vs. MRC in terms of packet error rate (PER) when the number of bytes per packet is 100, 500 and 1000.

mance among receiver diversity techniques (probability mass function shifted towards higher values), again in accordance with theoretical predictions.

Figure 13 shows the BER vs. SNR curve for SISO and MRC in the NLOS scenario. MRC provides a benefit vs. SISO of 1-2dB at intermediate SNR regimes, well within the theoretical upper/lower bounds of $3dB$ and $0dB$ for perfectly uncorrelated and perfectly correlated channels, respectively. Finally, Figure 14 shows the PER vs. SNR curve for SISO and MRC in the NLOS scenario with different packet sizes. The results reported in the figure show that MRC clearly outperforms SISO, especially with relatively larger packet sizes.

VIII. Conclusions and Future Work

In this paper, we have presented the MIMONet testbed featuring a dual software OFDM transceiver implementation operating on the same hardware. This dual implementation has achieved, for the first time to our best knowledge, a cross validation of the two implementations. Furthermore, we have presented the design and realization of a fine grained SNR and BER estimation methodology, that allowed us to carefully validate performance of the two software implementations against theoretical predictions.

The results of our study showed that, while both implementations in general behave according to theoretical predictions, Matlab implementation yields consistently lower BER values for comparable SNR ranges than those provided by GNU Radio. This is likely due to the DC offset effect, which is only partially solved in GNU Radio.

As future work, we want to address the DC offset problem experienced in the GNU Radio implementation enhancing the receiver design in such a way that the DC component is moved out of the band-of-interest. Furthermore, we are currently working on adding external clock support to MIMONet, so that MIMO and distributed MIMO techniques can be implemented using both software environments.

References

[1] I.F. Akyildiz, W.Y. Lee, M.C. Vuran, and S. Mohanty. A Survey on Spectrum Management in Cognitive Radio Networks. *IEEE Communications Magazine*, (4):40–48, 2008.

[2] A. Sibille, C. Oestges, and A. Zanella. *MIMO: From Theory to Implementation*. Academic Press, 2010.

[3] M. Dillinger, K. Madani, and N. Alonistioti. *Software Defined Radio: Architectures, Systems and Functions*. John Wiley & Sons, 2003.

[4] GNURadio. http://gnuradio.org/.

[5] S. Nanda and K.M. Rege. Frame Error Rates for Convolutional Codes on Fading Channels and the Concept of Effective Eb/No. *IEEE Trans. on Vehicular Technology*, 47(4):1245–1250, 1998.

[6] EttusResearch. http://www.ettus.com/.

[7] WARP. http://warp.rice.edu.

[8] Microsoft Research. http://research.microsoft.com/en-us/projects/sora/.

[9] K. Mandke, S.-H. Choi, G. Kim, R. Grant, R. C. Daniels, W. Kim, R. W. Jr. Heath, and S. M. Nettles. *Early Results on Hydra: A Flexible MAC/PHY Multihop Testbed*. 22–25 Apr., Dublin, Ireland 2007.

[10] W. Kim, O. Khan, K.T. Truong, S.H. Choi, R. Grant, H.K. Wright, K. Mandke, R.C. Daniels, R.W. Heath, and S.M. Nettles. An experimental evaluation of rate adaptation for multi-antenna systems. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 2313–2321, 19–25 Apr., Rio De Janeiro, Brazil, 2009.

[11] S. Gollakota, S. D. Perli, and D Katabi. Interference Alignment and Cancellation. In *Proc. ACM SIGCOMM*, pages 159–170, 2009.

[12] K. C. Lin, S. Gollakota, and D Katabi. Random Access Heterogeneous MIMO Networks. In *Proc. ACM SIGCOMM*, pages 146–157, 2011.

[13] W.L. Shen, Y.C. Tung, K.C. Lee, K.C. Lin, S. Gollakota, D. Katabi, and M.S Chen. Rate Adaptation for 802.11 Multiuser MIMO Networks. In *Proc. ACM Mobicom*, 2012.

[14] H. Yu, L. Zhong, A. Subharwal, and D. Kao. Beamforming on Mobile Devices: A First Study. In *Proc. ACM Mobicom*, 2011.

[15] E. Aryafar, N. Anand, T. Salonidis, and E. Knightly. Design and Experimental Evaluation of Multi-User Beamforming in Wireless LANs. In *Proc. ACM Mobicom*, 2010.

[16] K. Tan, H. Liu, J. Fang, W. Wang, J. Zhang, M. Chen, and G. M. Voelker. SAM: Enabling Practical Spatial Multiple Access in Wireless LAN. In *Proc. ACM Mobicom*, 2009.

[17] 802.11-2012 - Wireless lan medium access control (mac) and physical layer (phy) specifications, March 2012.

[18] T. M. Schmidl and D. C. Cox. Robust frequency and timing synchronization for OFDM. *IEEE Transactions on Communications*, 45(12):1613–1621, 1997.

[19] J. van de Beek et al. ML Estimation of Time and Frequency Offset in OFDM Systems. *IEEE Transactions on Si*, 45:1800–1805, 1997.

[20] J.Lee, H-L.Lou, and D. Toumpakaris. Joint Maximum Likelihood Estimation of Integer Carrier Frequency Offset and Channel in OFDM Systems. In *Proc. of the IEEE ICC 2006*, 2006.

[21] J. G. Proakis. *Digital Communications*. 3. edition, 1995.

[22] A. Viterbi. Convolutional Codes and Their Performance in Communication Systems. *IEEE Transactions on Communications Technology*, 19:751–772, October 1971.

[23] L. Messerschmitt. *Digital Communications*. Kluwer Academic Publishers, 3. edition, 1988.

[24] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 Packet Delivery from Wireless Channel Measurements. In *Proc. ACM SIGCOMM*, pages 159–170, 2010.

[25] http://www.mathworks.com/.