DIWINE

Contract No. CNET-ICT-318177

Evaluation of practical system constraints D5.12

Contractual date:	M21
Actual date:	M21
Authors:	David HALLS, William THOMPSON, Stefano GALIMBERTI, Stefano SAVAZZI, Tomáš HYNEK, Jan SÝKORA, Justin P. COON
Participants:	TREL, P+F, PdM, CTU, UOXF
Work package:	WP5
Security:	Public
Nature:	Report
Version:	1.0
Number of pages:	76

Abstract

Detailed assessment of all HW impairments and constraints of the demonstrator platform that affects the algorithms performance and should be reflected in the algorithm design; includes practical measurements of the HW induced effect on performance which should serve as feedback for the algorithm amendment.

Keywords

Synchronisation, scalability, processing delay, real-time, propagation, USRP, relay, WirelessHART, TSMP, mesh network, safety critical, industrial wireless.

Contents

Abbreviations

AF	Amplify-and-Forward
BER	Bit Error Rate
BSLD	Block Structured Layered Design
BSPK	Binary Phase Shift Keying
CA	Cloud Access
CAZAC	Constant Amplitude Zero Autocorrelation
CFO	Carrier Frequency Offset
CIMC	Critical Industrial Monitoring and Control
CIP	Cloud Initialisation Procedure
CP	Cyclic Prefix
CPU	Central Processing Unit
CRB	Cramér-Rao Bound
CRC	Cyclic Redundancy Check
CSE	Channel State Estimation
CSI	Channel Station Information
CSMA	Carrier Sense Multiple Access
DF	Decode-and-Forward
DLA	Distributed Learning Algorithm
DSSS	Direct-Sequence Spread Spectrum
FCS	Frame Check Sequence
FDMA	Frequency Division Multiple Access
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
GPS	Global Positioning System
GPSDO	GPS Disciplined Oscillator
GSM	Global System for Mobile Communications
GW	Gateway
HIS	Hierarchical Side Information
HNC	Hierarchical Network Codes
HW	Hardware
ISM	Industrial, Scientific and Medical
LDPC	Low Density Parity Check
LoS	Line-of-Sight
LQI	Link Quality Indicator
MAC	Multiple Access Channel; Media Access Control
MAC	Medium Access Control
MIMO	Multiple Input Multiple Output
MIPS	Microprocessor without Interlocked Pipeline Stages

MISO	Multiple-Input Single-Output
MSE	Mean Squared Error
MTU	Maximum Transmission Unit
NIC	Network Interface Card
NL	Network Layer
NSI	Network State Information
OFDM	Orthogonal Frequency Division Multiplexing
P2P	Peer-to-Peer
PER	Packet Error Rate
PHY	Physical Layer
PL	Payload
PLL	Phase Locked-Loop
ppb	part per billion
ppm	parts per million
PPS	Pulse Per Second
QoS	Quality of Service
RAM	Random Access Memory
RAT	Radio Access Technology
RF	Radio Frequency
RISC	Reduced Instruction Set Computing
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
SF	Synchronisation Frame
SIR	Signal-to-Interference Ratio
SLS	System-Level Simulator
SMN	Smart Meter Network
SoC	System on Chip
SPB	Samples per Block
SPP	Samples per Packet
SW	Software
TA	Timing Advance
TCXO	Temperature Compensated Crystal Oscillators
TDMA	Time Division Multiple Access
TL	Transport Layer
TO	Timing Offset
TSCH	Time Synchronised Channel Hopping
UDP	User Datagram Protocol
USRP	Universal Software Radio Peripheral
WPLNC	Wireless Physical Layer Network Coding
ZF	Zero-Forcing

1 Executive summary

This report evaluates the limitations and constraints of both the Smart Meter Network (SMN) hardware demonstrator and the Critical Industrial Monitoring and Control (CIMC) hardware demonstrator.

The first half the SMN section looks at limitations relating to the WPLNC algorithms. A sampling rate of 1 MS/s is chosen along with the use of Orthogonal Frequency Division Multiplexing (OFDM) with 64 bins at a centre frequency at 2.4 GHz. The DIWINE air interface is updated, including the design and layout of the PiCSE and PiHRC pilots. Some testing is also performed. The half-duplex constraint is discussed and the processing delay of the system is shown to be small enough to allow real-time operation with only basic baseband processing. Once the processing of either the DIWINE algorithms is included, however, the processing delay becomes too long and the system runs in a quasi-real-time fashion. The maximum length of packets is discussed and the multiple access strategy for the final gateway nodes to the destinations is chosen as Time Division Multiple Access (TDMA). Time of flight is shown not to be an issue and there will be minimal asynchronism in the system in the external synchronisation case, and the use of OFDM will mitigate the impact. The system under test is detailed and a maximum of a 9-node setup will be implemented. Node visibility is highlighted as being a minor issue that affects one of the algorithms, but can be circumvented using message passing.

The second half of the SMN section details the hardware limitations relating to the synchronisation phase of the network. This looks into the frequency stability and suitability of two reference oscillators. The modifications of the algorithm to meet the half duplex constraint are also discussed, as well as the need to use a higher sampling rate in the synchronisation phase of 4 MS/s to improve the offset estimation accuracies. Limits on the maximum separation of connected nodes are also detailed based upon the signals' time-of-flight. Discussion on the impact on the limitations and constraints, were an application specific hardware solution to be developed, concludes the SMN section.

The CIMC hardware demonstrator limitations and constraints are highlighted in this report with respect to the relevant industrial scenario and application cases. The material presented in this deliverable provides a detailed description of the demonstrator hardware and software capabilities and provides guidelines to tailor the DIWINE algorithms and minimise implementation-related issues. The CIMC demonstrator implements a dual-processor, and dual radio access architecture, with one section dedicated to the support of DIWINE-identified algorithms while the other ensures the compatibility with the industry standard WirelessHART. The two radio modules are designed to cooperate during synchronisation and topology control sessions. In particular, the DIWINE section ensures a flexible PHY and MAC layer implementation. It provides full support to industry-standard IEEE 8021.5.4 air-interface with extended functionalities to allow for good real-time capability, low energy consumption and software-defined PHY transmission modes. The deliverable highlights the main CIMC expected hardware and software limitations and constraints, with respect to the low-power core processor processing power, PHY, MAC and application layer limitations for target algorithms/scenarios and interference limitations in unlicensed spectrum sharing scenarios.

2 Smart meter network hardware demonstrator

2.1 Introduction

This section of the report provides details on the Smart Meter Network (SMN) hardware demonstrator and evaluates the associated limitations and constraints. The SMN hardware demonstrator is based on the Universal Software Radio Peripheral (USRP) hardware platform combined with the GNU Radio software platform. The SMN hardware demonstrator is made up of a network of USRPs each connected to a ‘host’ computer running GNU Radio which is an open-source software environment. The contents of the report are vital for the design of the algorithms in WP3 and WP4, such that the algorithms are run within the inherent constraints of the hardware testbed.

The section devoted to the SMN is split into two subsections, devoted to the two main areas of the DIWINE paradigm that will be implemented: Wireless Physical Layer Network Coding (WPLNC) and distributed network synchronisation. The first looks at the limitations and constraints of the Hardware (HW) testbed that are specific to WPLNC. It is decided that a sampling rate of 1 MS/s is sufficient for the implementation of the algorithms, and allows 2 USRPs to be controlled from one laptop. The use of Orthogonal Frequency Division Multiplexing (OFDM) with 64 bins at a centre frequency in the Industrial, Scientific and Medical (ISM) band at 2.4 GHz is also appropriate. Successful testing of the DIWINE pilots is shown. The half-duplex constraint is reinforced; both of the WPLNC algorithms that are to be implemented adhere to this restriction. Once the processing of either WPLNC algorithm is included in the system, the processing delay becomes too long and the system runs in a quasi-real-time fashion.

The length of packets is restricted by buffer sizes in GNU Radio, however they are also limited by the necessity for regular synchronisation frames, so the limit caused by GNU Radio is not a problem. It is shown that the time of flight issue will not affect the system in the indoor testing environment. There should also be minimal asynchronism in the system, especially with the aid of external synchronisation, and using OFDM should mitigate any issues. The system under test entails a maximum of a 9-node setup. Node visibility will be highlighted as being a minor issue that affects one of the WPLNC algorithms but it can be circumvented by enabling message passing in GNU Radio.

The second half of the SMN section looks at limitations related to the distributed synchronisation algorithm. Hardware impairments of the USRPs focuses on the reference oscillators. It is shown that the internal oscillator suffers from significantly more phase drift and frequency instability compared to Ettus’ GPSDO. The benefits of using the GPSDO as the reference oscillators is shown, highlighting that the objective of the testbed should be to demonstrate the core DIWINE principals rather than be a final implementation solution. Due to the half-duplex constraint, the scheduling of the synchronisation algorithm has been adapted so that all nodes synchronise to multiples of 64 µs, rather than a single global time. Other modifications to improve the algorithm are discussed, such as the increase in symbol rate and windowing the data to cope with non-complete capture of synchronisation frames.

The limitations of the system would be relaxed/improved somewhat if an application-specific implementation were used as detailed in Section 2.5. These include the host processing (which would not be required at all), processing delay and packet lengths. The system under test would expand which could lead to time-of-flight and asynchronism issues as well as a change in the node visibility issues. Parameters such as the bandwidth, the half-duplex constraint, the use of OFDM in the ISM band, and the DIWINE air interface would not change, however. With regard to the distributed synchronisation algorithm, the limitations would remain mostly unchanged, although the use of a device with a higher quality crystal oscillator would significantly improve performance.

2.2 Wireless physical layer network coding related constraints

2.2.1 Bandwidth limitations, host processing, multiplexing and centre frequency

As reported in [1] an OFDM solution is employed with $N_{FFT} = 64$ subcarriers. The initial OFDM parameters, including the initial sampling rate are shown in Table 2. The use of OFDM offers scalability, whereby the system bandwidth and Fast Fourier Transform (FFT) size could theoretically be scaled up from these values, although the current values are thought fit for purpose. CTU concur with TREL that since the proposed DIWINE algorithms aim to achieve ‘proof-of-concept’ the bandwidth of current implementation is sufficient for all the proposed algorithms to show their properties.

Parameter	Value
FFT size, N_{FFT}	64
Cyclic prefix length	16
Sampling rate	1 MS/s
Centre frequency, f_c	2.4 GHz

Table 1: OFDM parameters

The current use of a sample rate of 1 MS/s is deemed sufficient to provide enough bandwidth for proof-of-concept of the two WPLNC algorithms: the Block Structured Layered Design (BSLD) and the Distributed Learning Algorithm (DLA). Using this sample rate allows two USRPs to be reliably controlled from one i7 Toshiba laptop. In the 2×1 Multiple-Input Single-Output (MISO) scenario (see Figure 1) controlling the two source USRPs from one laptop, and the destination USRP from another laptop, allows reliable free running. However, attempting to run all three USRPs from the same laptop causes occasional glitches in the received signal due to lack of resources. This results in a distorted received signal as illustrated in Figure 2, where the plot *should* show a continuous OFDM waveform as is present at the beginning and end of the trace.

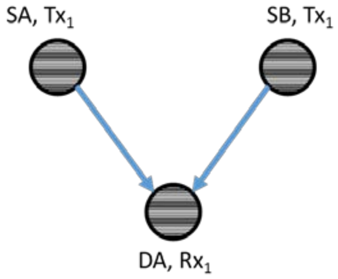


Figure 1: 2×1 MISO scenario

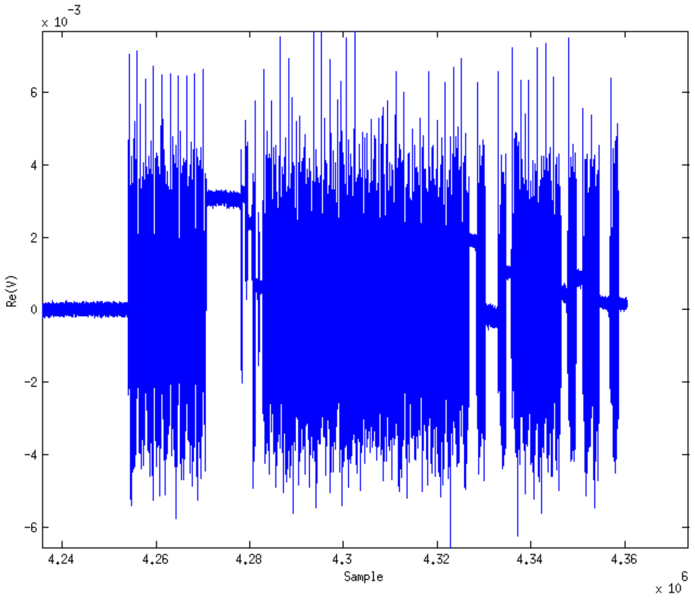


Figure 2: Received waveform with insufficient resources

The reason for this problem is highlighted by the data in Table 2 which shows the approximate processor load of the receiver and transmitter laptops obtained using the ‘top’ command in Linux. The laptops have quad-core processors, so a total load of 400% is achievable and using the ‘System Monitor’ tool, it was seen that load balancing across the cores is achieved effectively in that each core was running ¼ of the total load. The two numbers in the receiver (Rx) column in the table indicate when the receiver is running but idle and when it is receiving, i.e. processing data. It can be seen that even at 1 MS/s that the total load is approximately 375% if both transmitter and receiver were running on the same machine. This is not far below the available 400% showing why occasionally, during peak demand, resources are exhausted and glitches in the received waveform occur. The table also shows why sample rates higher than 1 MS/s are not reliably supported because the total required resources exceeds 400%. In the 4 MS/s case, there are not even enough resources for the receiver to run on its own when it is receiving data. Using a desktop i7 PC with an external Gigabit Ethernet port, it was found that all three USRPs could be run *more* reliably from the one machine, even though 400% of the resources were constantly being used.

Sample Rate (MS/s)	Rx (idle % / active %)	Tx (%)
1	200/250	125
2	225/350	175
4	335/–	225

Table 2: Computational load

Although the processing power of the computers limits the number of USRPs that can be controlled by each laptop, the network capacity is not so restrictive. Each USRP requires 32 b to transfer each 16 b I/Q baseband message. This means that at 1 MS/s only 32 Mbit/s of Ethernet bandwidth is required by each USRP. As a result, up to 30 USRPs can be serviced by one Gigabit switch at 1 MS/s. This gives great flexibility in that up to 30 USRPs, i.e. a number in excess of the system under test in the DIWINE (see Section 2.2.8), can all be connected to the same wired switch to receive and transmit their baseband data.

At a sample rate of 1 MS/s the cyclic prefix of 16 samples will be 16 µs, this is sufficient to avoid ISI in the indoor environment in which testing will take place where delay spreads will be of the order of nanoseconds. The initial 64 carrier OFDM system is also deemed sufficient for the WPLNC algorithms. Of the 64 subcarriers in each OFDM symbol there are currently 48 data subcarriers, a DC carrier, 11 guard carriers and four pilot carriers at -21, -7, 7 and 21. The pilot implementation is commensurate with IEEE 802.11 as shown in the top of Figure 3. These pilots, available in each payload symbol, can be used for channel estimation and tracking. In the 2×1 MISO system the pilots are orthogonalised as shown in the bottom of Figure 3, by allocating two of the four pilots to each source. This implementation clearly has limited scalability and although it can be used for the 2-relay butterfly (see Figure 4) where there are two nodes per stage, the more advanced scenarios will require non-orthogonal network synchronisation (see Section 2.3). Where external synchronisation *is* available, and the channel is relatively static these ‘tracking’ pilots are not utilised. In the absence of external synchronisation, however, they are used even when the channel is relatively static to correct for residual Carrier Frequency Offset (CFO) after estimation (as detailed in Section 1.2.7 of [1]).

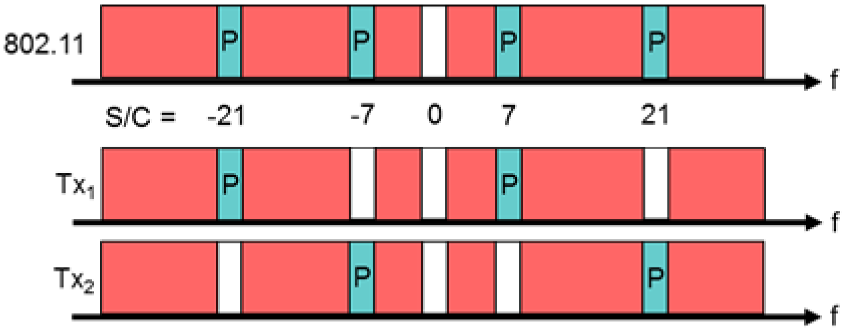


Figure 3: Pilot structure for point-to-point 802.11 (top) and 2×1 MISO (bottom) systems

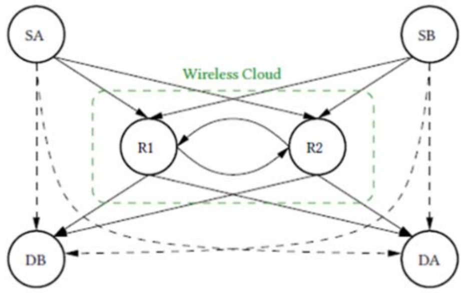


Figure 4: 2-relay butterfly without perfect intra-cloud coordination (S-RC-b)

The centre frequency of 2.4 GHz is suitable for the algorithms under test. Testing has been successfully performed simultaneously at 2.40, 2.45 and 2.48 GHz to allow coexistence of different algorithms and scenarios in the same testing environment. These frequencies have the advantage that they are in the ISM band so no special license is required, however, it suffers from the potential disadvantage of co-channel interference. This has not been found to be a problem in the TREL testing environment where there are no co-existing devices at 2.40, 2.45 or 2.48 GHz.

2.2.2 Air interface, channel estimation and DIWINE pilots

The frame structure for the initial point-to-point testing is shown in Figure 5. The preamble is made up of two synchronisation symbols followed by a header, and then the payload symbols. The two synchronisation symbols were used for timing synchronisation and coarse and fine frequency estimation as detailed in Section 1.2.5 of [1]. The header symbol contains information including packet number, payload length and modulation scheme.

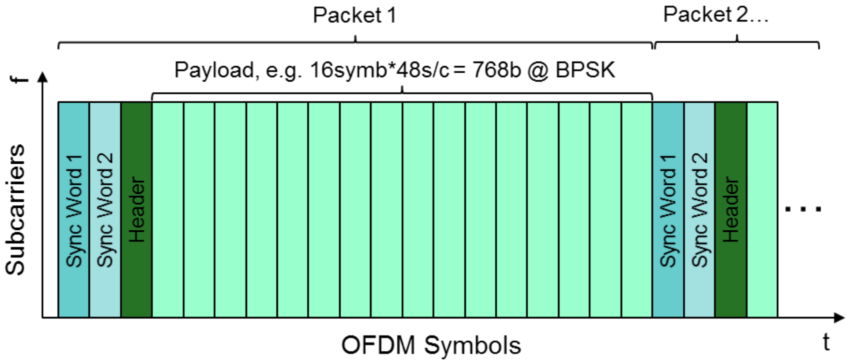


Figure 5: Point-to-point frame structure

This was then expanded to work for the 2×1 MISO system, which is the smallest possible building block of the overall system as shown in Figure 6. In the 2×1 MISO system, in order that the synchronisation and channel estimation can be performed independently between Tx_1 (*SA*) and *DA*, and Tx_2 (*SB*) and *DA*, the preambles are transmitted orthogonally in time, as shown in Figure 7. As previously mentioned, in this scenario the pilots are also orthogonalised as shown in the bottom of Figure 3, by allocating two of the four pilots to each source, but this is not scalable. In the future systems, if these pilots are not required due to the DIWINE distributed synchronisation techniques, they can be replaced by data subcarriers.

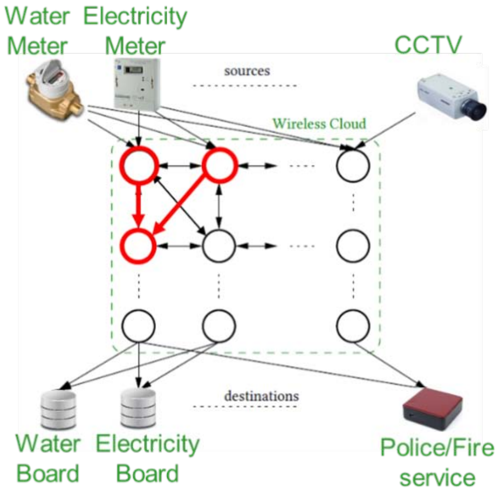


Figure 6: SMN vision

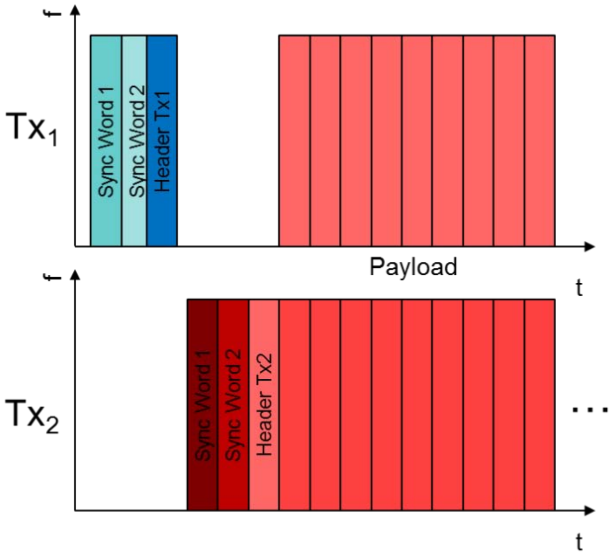


Figure 7: 2×1 MISO frame structure

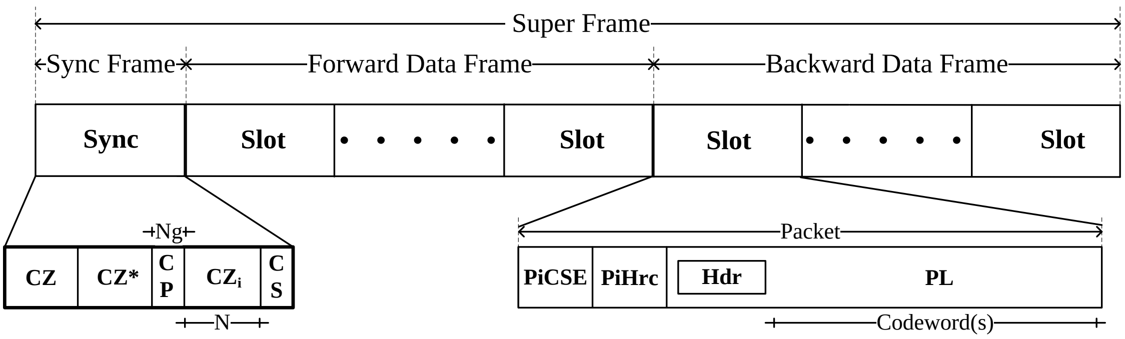


Figure 8: DIWINE super-frame structure

As an interim step to implementing the full DIWINE super-frame structure, shown in Figure 8[1], testing has been performed by incorporating the DIWINE specific pilots (PiCSE used for Channel State Estimation (CSE) and PiHRC used for hierarchical information) into the Payload (PL) section of the existing 2×1 frame structure (Figure 7) thus resulting in a PL as in Figure 9. This means that the existing preamble can be used for CFO and TO estimation and correction (as well as using the existing channel estimation scheme as a benchmark to test the performance of the CSE using PiCSE).

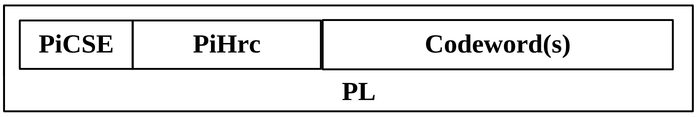


Figure 9: Interim super-frame structure

The length of the PiCSE and the codeword(s) sections, in terms of number of OFDM symbols, can be varied. The length of PiHRC is currently fixed as 16 OFDM symbols (each containing 2 bits mapped to a Quaternary Phase Shift Keying (QPSK) constellation) with the format shown in Table 3.

Preamble	S_1 Idx	S_2 Idx	S_3 Idx	S_4 Idx	HNC Idx	Pad	CRC
0011	XXX	XXX	XXX	XXX	XXXXX	XXXXXXX	XXXX

Table 3: PiHRC structure

In order to allow the PiHRC to be transmitter simultaneously from multiple sources, the data mapped to each of the 16 OFDM symbols, i.e. each pair of bits once mapped to a QPSK constellation, are spread along the 48 data subcarriers in frequency using a Constant Amplitude Zero Autocorrelation (CAZAC) sequence of length 47 with root 1, padded with a 0. This use of an odd sequence length provides beneficial cross-correlation properties. The CAZACs used for this spreading also uniquely identify the node transmitting the PiHRC by using a cyclic shift equal to the index of the node, i.e. 0 for S_1 (stored as 000 in PiHRC) and 1 for S_2 (stored as 001 in PiHRC). The index 101 in PiHRC indicates a blank field and was chosen to minimise PAPR issues resulting from long strings of 0. The 5 bits for the HNC index are currently being defined by WP4. The padding is a random series of 7 bits, this may be used at a later date if more information needs to be included in PiHRC. The final 4 bits are a Cyclic Redundancy Check (CRC) with polynomial $x^4+x^3+x^2+x$.

Due to the fact that the channel is relatively static over time in the laboratory environment, but timing offsets cause a phase rotation as a function of frequency leading to an apparently non frequency flat channel, the PiCSE are implemented as spreading codes along time. Thus the same CAZAC is repeated over all 48 data subcarriers. This is illustrated in Figure 10 which shows the time-frequency layout of PiCSE and PiHRC. The length of the PiCSE CAZAC can be varied and is given by N_{PiCSE} in Figure 10.

[1] Note the removal of PiAcq, compared to previous versions, as this is achieved by dint of the CAZAC sequences used in the PiCSE instead, and the reordering of PiCSE and PiHRC for convenience.

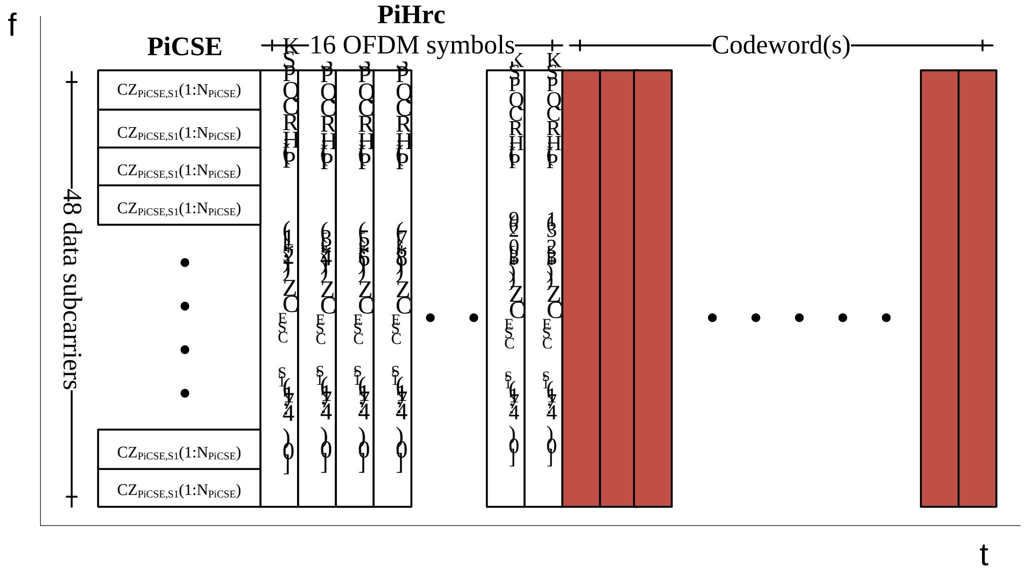


Figure 10: Time-frequency frame structure of interim air interface

Testing of the performance of PiCSE was undertaken by transmitting over the 2×1 system. Firstly the CSE is obtained from the preamble. The channel estimation is performed in the frequency domain; it is performed after the FFT and the Cyclic Prefix (CP) removal. A wideband estimate can be obtained from the second synchronisation symbol in the preamble (‘Sync Word 2’ in Figure 7, abbreviated to ‘*sync*’ in the following). The symbol transmitted from S$_1$ in the frequency domain is $X_{S_1}^{sync}[k]$, where $k = 0, \dots, N_{used} - 1$ and N_{used} is the number used subcarriers, 48 in this case. The synchronisation symbol is a pseudo-random sequence as defined in [2] such that $X_{S_1}^{sync}[k] \in \{1, -1\}$ for all data subcarriers, and 0 elsewhere. The received symbol in the frequency domain from the second OFDM symbol, i.e. ‘Sync Word 2’ from S_1, of the first of the two preambles is given by $\tilde{Y}^{sync}[k, 2]$. Channel estimation of $\widehat{H}_{S_1}^{sync}[k]$ is performed by:

$$\widehat{H}_{S_1}^{sync}[k] = \frac{\tilde{Y}^{sync}[k,2]}{X_{S_1}^{sync}[k,2]}. \tag{1}$$

At a high SNR of approximately 35 dB, achieved with the transmit and receive USRPs in close proximity and a transmit gain of 20 dB, an accurate CSE can be assumed to be obtained in this fashion.

This can then be compared against the CSE obtained from PiCSE. This is obtained per subcarrier k by taking the dot product of the N_{PiCSE} received symbols with the CAZAC$_{\mathrm{PiCSE}}$ of the source(s) that are expected to be present in the signal. For example:

$$\begin{aligned}\widehat{H}_{S_1}^{PiCSE}[k] = \big(&\tilde{Y}^{PiCSE}[k,1].\,CZ_{PiCSE,S_1}[1]^* + \tilde{Y}^{PiCSE}[k,2].\,CZ_{PiCSE,S_1}[2]^* + \cdots \\ &+ \tilde{Y}^{PiCSE}[k,N_{PiCSE}].\,CZ_{PiCSE,S_1}[N_{PiCSE}]^*\big)/N_{PiCSE.}\end{aligned} \tag{2}$$

The performance of PiCSE relative to the CSE obtained from the preamble for different N_{PiCSE} are shown in Table 4. The 'relative power' indicates the transmit power of the PiCSE symbols relative to the preamble symbols power. The channel values were normalised by the mean channel powers, and the Mean Squared Error (MSE) between the channel estimate from PiCSE and from the preamble was calculated and is shown in the table. It can be seen that as the transmit power of the PiCSE is reduced, the MSE increases. More importantly it can be seen that using shorter N_{PiCSE} does not increase the MSE significantly at high PiCSE transmit power, but has it does at lower transmit powers of PiCSE.

	N_{PiCSE}		
Relative Power (dB)	**4**	**8**	**16**
0	9.2e-4	8.5e-4	7.8e-4
-10	1.2e-3	9.2e-4	8.7e-4
-20	5.2e-3	2.7e-3	1.8e-3
-30	4.0e-2	2.4e-2	1.0e-2
-40	3.4e-1	3.2e-1	8.2e-2

Table 4: MSE between preamble CSE and PiCSE

Once the CSE has been obtained from PiCSE this can be applied to the received PiHRC symbols using a simple Zero-Forcing (ZF) equaliser:

$$\hat{R}_{S_1}^{PiHRC}[k,t] = \frac{\tilde{Y}^{PiHRC}[k,t]}{\hat{H}_{S_1}^{PiCSE}[k]}. \tag{3}$$

where $t = 1, \dots, N_{PiHRC}$. The transmitted QPSK on each OFDM symbol of the PiHRC can then be gleaned at the receiver. This is obtained from the dot product of the CAZAC$_{\text{PiHRC}}$ of the node that transmitted the PiHRC with the $N_{used} = 48$ equalised symbols, at each OFDM symbol. For example the QPSK symbol on the first OFDM symbol of PiHRC transmitted by S_1 is given by:

$$\begin{aligned} PiHRC_{S_1}[1] = \big(&\hat{R}_{S_1}^{PiHRC}[1{,}1].\,CZ_{PiHRC,S_1}[1]^* + \hat{R}_{S_1}^{PiHRC}[2{,}1].\,CZ_{PiHRC,S_1}[2]^* + \cdots \\ &+ \hat{R}_{S_1}^{PiHRC}[N_{used},1].\,CZ_{PiHRC,S_1}[N_{used}]^*\big)/N_{used.} \end{aligned} \tag{4}$$

and the N_{PiHRC}-*th* QPSK symbol is given by:

$$\begin{aligned} PiHRC_{S_1}&[N_{PiHRC}] \\ &= \big(\hat{R}_{S_1}^{PiHRC}[1, N_{PiHRC}].\,CZ_{PiHRC,S_1}[1]^* \\ &+ \hat{R}_{S_1}^{PiHRC}[2, N_{PiHRC}].\,CZ_{PiHRC,S_1}[2]^* + \cdots \\ &+ \hat{R}_{S_1}^{PiHRC}[N_{used}, N_{PiHRC}].\,CZ_{PiHRC,S_1}[N_{used}]^*\big)/N_{used.} \end{aligned} \tag{5}$$

These N_{PiHRC} (nominally 16) QPSK symbols can then be mapped to the 32 bits of the PiHRC transmitted by S_1. The same equalisation as equation (3) is also performed on the codewords section of data.

Initial testing using this PiCSE, PiHRC structure and populating the codewords with those generated by the DLA (see Figure 11 for node diagram and Section 2.2.2 of [1] for algorithm details) has begun. So far this includes successful implementation of both source and relay MATLAB code running online as compiled shared libraries within GNU Radio blocks. The relay(s) to destination transmission, and the relay decoding is currently performed offline in MATLAB. Using $N_{PiCSE} = 4$ was sufficient to achieve 0 BER from end-to-end in the high SNR testing environment in the laboratory (about 35 dB). This will therefore be used as the initial baseline value.

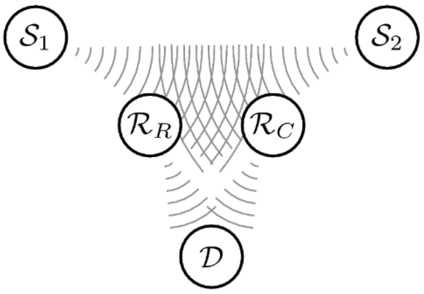


Figure 11: 2-source, 2-relay, 1-destination scenario for distributed learning process algorithm

Since the performance of WPLNC seems to be strongly dependent on channel parameterisations, the quality of the channel estimation is an important factor [3][4]. The impact of channel parameterisation as well as quality of its estimation on the capacity was intensively studied in [5]. Therein a single carrier modulation system in a butterfly network configuration is studied. The results show how the symmetric sum-rate capacity is affected by incorrect channel estimation (expressed by the variance of estimation error σ) for various Hierarchical Network Codes (HNCs), channel parameters and modulation schemes in a two-source one-relay network. The results in the following figures show the dependence of the symmetric Multiple Access Channel (MAC) capacity on the variance of the estimation error σ_Δ^2 for Binary Phase Shift Keying (BPSK) and QPSK alphabets. In the case of QPSK, different HNCs are assumed. The true channel parameterisation is assumed to be $h_i = \hat{h}_i + \Delta h_i$. It can be stated from the obtained results that as long as the estimation variance is lower then 0.01 it does not have disastrous effect on the channel capacity regardless of the channel parameterisation and HNC function. In [6] they develop a low-complexity channel estimation technique that can achieve the Cramér-Rao Bound (CRB) at high SNR. Compared with a WPLNC scheme with perfect Channel Station Information (CSI), their channel estimation technical has only a 0.5–1 dB loss in end-to-end Bit Error Rate (BER) performance at high SNR. These single carrier results can be straightforwardly extended to the current OFDM implementation.

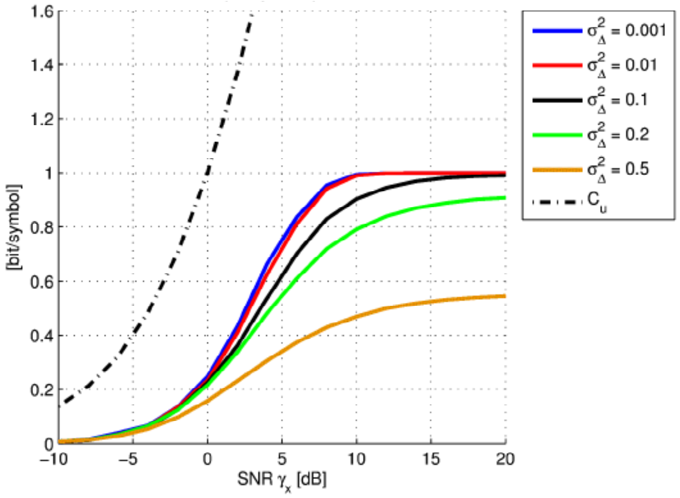


Figure 12: MAC capacity for BPSK $h_A = h_B = 1$ [5]

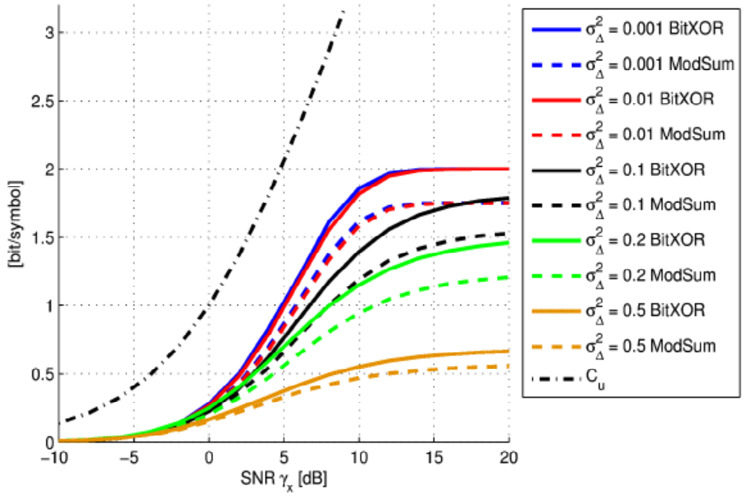


Figure 13: MAC capacity for QPSKa $h_A = h_B = 1$ [5]

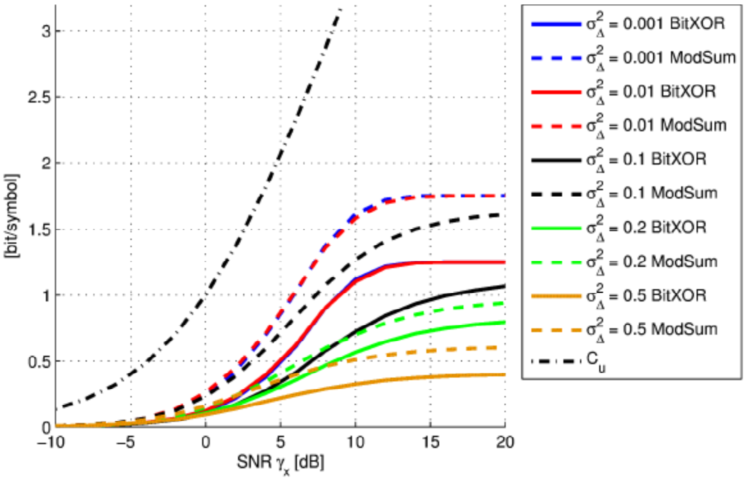


Figure 14: MAC capacity for QPSKa $h_A = 1$, $h_B = j$ [5]

2.2.3 Half-duplex constraint

The half-duplex constraint is a natural issue that arises with wireless nodes. It forces the communication in the network to be slotted because each node is able to only receive or transmit in any given time instant. All of the proposed WPLNC algorithms take this constraint into account, in order to match the definition of the random connectivity class. The SMN fits into the random connectivity cloud class which assumes random and sparse connectivity and no global Network State Information (NSI) knowledge, as well as a half-duplex constraint [7]. Also the proposed structures of the DIWINE air interface protocol (Figure 8) also respect this limitation. Each node is assigned time slots in which it transmits and it receives in the other slots. The other limitation that emerges from the half-duplex constraint is the time necessary to switch the USRP from receiving to transmitting. The turnaround time of the XCVR2450 daughterboards has not been characterised but Ettus Research has made direct assurances to TREL that it is <10 µs, which pales into insignificance when compared to the processing delays detailed in Section 2.2.4.

2.2.4 Processing delay

The processing delay in the relays is a very important factor in the HW demonstrator, particularly given the half-duplex constraint. There are a number of parts to the delay: reception and down-conversion, transfer to the laptop (Ethernet protocol related delays), baseband processing performed in GNU Radio on the laptop, e.g. decode/re-encode, then transfer back to the USRP, and finally up-conversion and transmission. It is difficult to accurately profile these individual components.

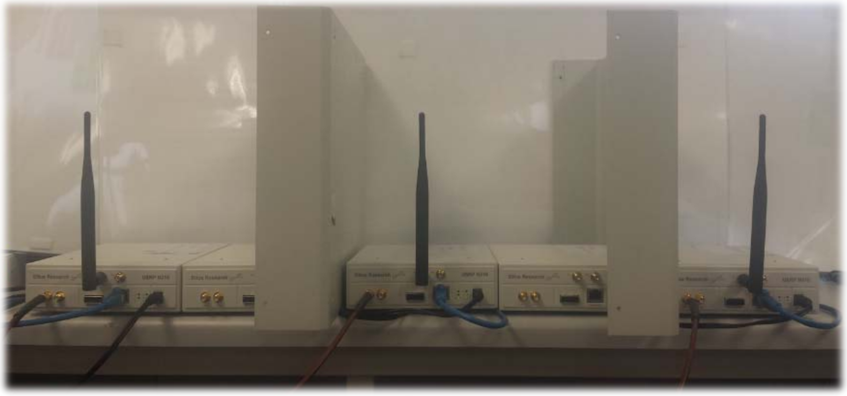


Figure 15: Single hop relay. From the left: source, relay and destination

It has been found that there is significant latency between the USRP and the host processor, and thus even in a simple Amplify-and-Forward (AF) relay system (shown in Figure 15), which contains only a simple packet detector and gain blocks in the baseband processing flow graph, has a total processing delay of nearly 5 ms. The total processing delay in the node also includes the time taken for data to travel from the USRP to the host processor and then from the host processor back to the USRP.

In the current settings for the relay setup shown in Figure 15, a large gap of 7 ms (τ_t) is left in between the beginning of the first time slot (when the source transmits) and the beginning of the second time slot (when the relay transmits) to allow for transmission of the packet *and* the delay in the relay ($\tau_{d'}$, which also includes the negligible propagation delay). This is wasteful as the packet fills only 1.52 ms (τ_p) of the 7 ms timeslot, as shown in Figure 16. This is the 19 OFDM symbols (3 preamble and 16 data), which are each 80 samples long (64 point FFT with ¼ CP), at a sampling frequency of 1 MS/s.

If the processing delay in the relay becomes so long that $\tau_d > \tau_t$ and the relay is forced to transmit after the allotted time slot has started, the USRP reports 'L' indicating that the transmit 'burst' has arrived at the USRP late, i.e. after the allotted transmit time, and the burst is dropped. If τ_t is reduced below 7 ms, occasionally late packets are reported, and below 5 ms all packets are reported as late. This shows that there is significant variation in the processing delay caused by buffering in the laptop and the USRPs and the Ethernet link between the laptop and the USRPs. Furthermore, this is thus already quasi-real-time system, and this is before any significant baseband processing is introduced at the relay!

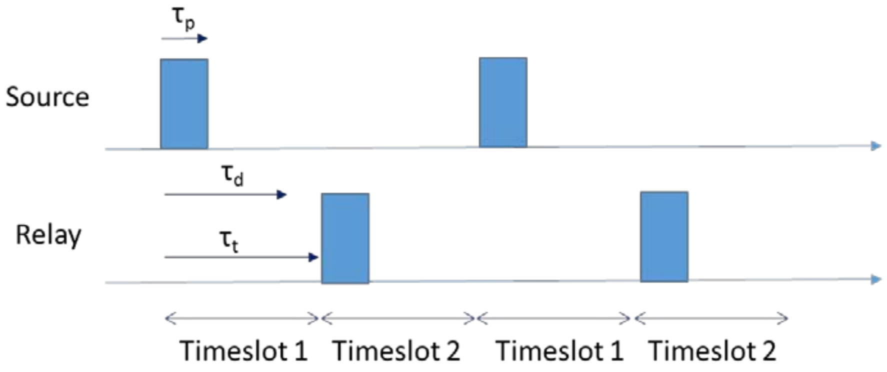


Figure 16: Processing delay example

The next stage of testing was to implement a Decode-and-Forward (DF) relay. As an interim step towards the BSLD implementation, this relay is the receiver from the 2×1 system (Figure 7), with OFDM modulation and transmission functionality added to the end of the flow graph. It should be highlighted that as there is no HNC mapping performed, at this stage, this is not a viable system but simply a flow graph to help test processing delay. It was found setting τ_t is above 7 ms resulted in no late packets, thus the total processing delay is very similar to that of the basic AF system. This implies that the baseband processing, in GNU Radio, is a very small element of the total processing delay, $\tau_{d'}$.

The Boost library boost::datetime has a microsecond precision clock (although its *accuracy* depends on the platform), from which the function 'gettimeofday' can be used to get microsecond precision timestamps. Adding one timestamp to the point in the GNU Radio code where the first sample of preamble is detected, and one to the final function before transmission, the total baseband processing delay can be estimated. It was found that the total delay through the flow graph was approximately 1 ms. This explains why the total processing delay in the AF and DF cases appear similar; the delay is in other parts of the system not the GNU Radio baseband processing, e.g. the laptop and USRP buffers, and Ethernet transmission.

There are a number of USRP parameters than can be adjusted which impact the processing delay, excluding the GNU Radio baseband processing: Samples per Block (SPB), Samples per Packet (SPP), and the receive frame size. The general guidelines are that operating at a higher sample rate will mean data will get to/from the USRP quicker. However, as reported previously there are limitations on this due to host processing power. Choosing a smaller SPB will mean less time filling your buffer on a 'recv' call at the receive USRP, so this is advantageous. In addition choosing a smaller SPP will mean the device will spend less time filling a packet's payload before sending it down the wire. It is also possible to adjust the Network Interface Card (NIC) Maximum Transmission Unit (MTU) size and the NIC interrupt rate. More details are investigated in [8], and the latency results of the latency study are shown in Figure 17. It can be seen that reducing the SPB down to 32 and SPP down to 64 provides the best performance at 1 MS/s. Unfortunately this did not discernibly reduce the overall processing delay and in order to avoid late packets τ_t has to remain above 7 ms. Ettus Research have confirmed that when looking at end-to-end latency, rather than just USRP to NIC, 5–10 ms is to be expected.

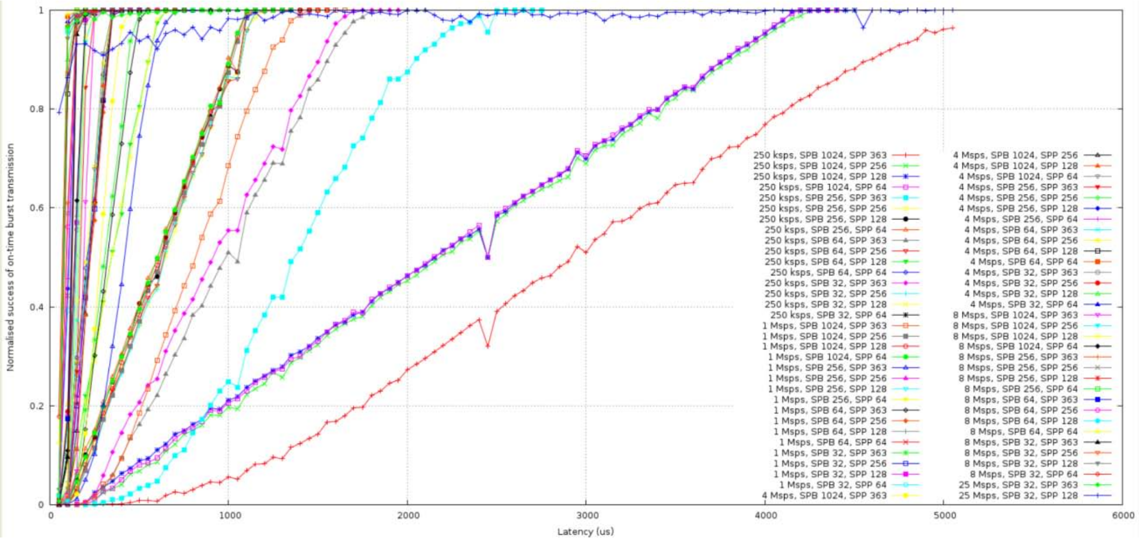


Figure 17: USRP latency results [8]

One simple way to improve the efficiency, and circumvent the effects of buffering and transfer by Ethernet (and the GNU Radio baseband processing delay), is to increase the number of packets sent in each burst. Transmitting one packet in the MISO case where two preambles means a longer τ_p. This is because there are 16 OFDM data symbols (80 samples each) and 6 preamble symbols which gives 1760 samples, i.e. 1.76 ms at 1 MS/s. Sending 10 packets per burst results in 17600 samples which means τ_p = 17.6 ms at 1 MS/s. It was found that increasing τ_t to 17.6 ms enables successful transmission and reception thus enabling a fully real-time system, as illustrated in Figure 18. In this case $\tau_p = \tau_{t,} > \tau_d$, and so the processing delay becomes irrelevant. The received time-domain waveform at the destination for four successive bursts, consisting of alternating bursts received from the source (smaller envelope) and the relay (larger envelope). The plot shows the lack of gaps between bursts, and therefore fully real-time operation.

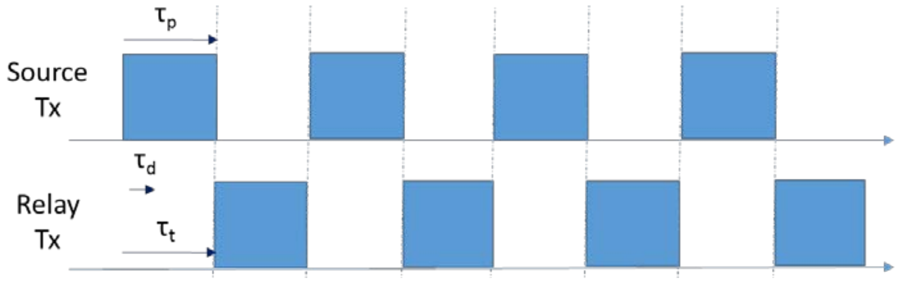


Figure 18: Real-time operation

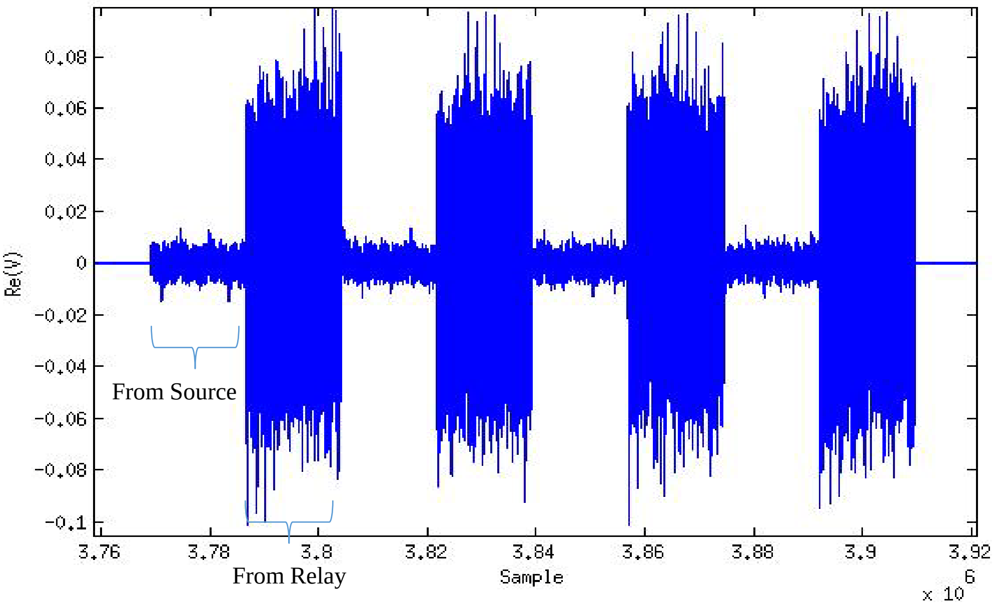


Figure 19: Received time domain waveform at destination for four successive transmit bursts

When implementing the BSLD algorithm, i.e. including the BSLD decoder in the DF relay code, the BSLD's baseband processing dramatically increases the total processing delay at the relay. In particular the Low Density Parity Check (LDPC) decoder takes a significant time to execute. Using the timing function in Python is was possible to determine the time taken by the various parts of the BSLD implementation. The algorithm was set such that 10 (GNU Radio) packets are transmitted in each timeslot, as in the previous DF testing. These are then processed in one execution, in order to improve the efficiency of the LDPC decoder. This means 7680 data bits per burst, made up of 17600 samples or 17.6 ms at 1 MS/s. The BSLD algorithm takes up to 250 ms in total, with 210 ms of that taken up by the LDPC decoder. The LDPC decoder is implemented in Cython (compiled C-code called from within Python) in or to give speed-up over a pure Python implementation. This long execution time means that the scheme cannot run in real-time, because the delay associated with the BSLD algorithm alone is significantly longer than the length of the source transmission burst. Some further optimisation of the code may be achieved, but this is unlikely to be sufficient and increasing the burst length is limited by buffering in GNU Radio. It is therefore concluded that this algorithm will have to run in a quasi-real-time fashion as shown in Figure 16.

The DLA is implemented in MATLAB, which is compiled as a shared library and run from within a GNU Radio block, and the time taken for the processing of each packet (as defined in Figure 10) at the relay is approximately 40 ms. In the current, initial, testing there are 12 OFDM symbols of codewords, i.e. payload, each of which contains 48 data subcarriers, with a total of 4 bits (2 from each source) per subcarrier. That results in 2304 bits per packet. If one ignores all other components of the processing delay for now, i.e. setting $\tau_t = 40$, this would result in an effective data-rate of 57.6 kbit/s. This will be significantly improved by optimising the MATLAB code, including improving efficiency by processing multiple packets per execution as with BSLD or rewriting some code as mex files or in C-code. Extending the length of the payload (see Section 2.2.5) and reducing the overheads, e.g. condensing PiHRC by removing the empty fields, or increasing the number of data subcarriers by removing the pilots or squeezing the guard bands, will also help improve efficiency. The effective data-rate can therefore be significantly improved but the algorithm will likely not be able to run in real-time. However, a proof-of-concept demonstration of a novel and highly complex algorithm will be achieved.

2.2.5 Length of payload

As mentioned in the previous section, the relative lengths of each section is variable but the initial values are $N_{PiCSE} = 4$, and $N_{PiHRC} = 16$. Due to buffer limits in the multiplexers in GNU Radio which joins the PL section to the existing 2×1 preamble before transmission, the total number of OFDM symbols in the PL is limited to 45. This means that the current maximum $N_{PL} = 25$. Fairly simple system redesign would allow extension of this by simply multiplexing multiple PL sections per preamble. However, without external synchronisation the impact of CFO and carrier frequency drift might limit the length of PL, considering that as can be seen from Figure 8 there will be a number of ‘slots’ in between each DIWINE synchronisation frame. See Section 2.3 for more details on CFO and carrier frequency drift.

2.2.6 Multiple access strategy from gateway to destination nodes

On the final stage of communication, the information must be delivered from cloud gateway relays to the final destination nodes, e.g. R_R and R_C to D in Figure 11, or the final row of relays to destinations in Figure 20 (although the latter is outside of the scope of the HW demonstrator). When the destination is connected to more than one relay then it has to reliably receive the signals from all of them. Due to the lack of Hierarchical Side Information (HSI) during this stage, the information must be delivered in some classical way. The first option is an orthogonal multiplexing scheme such as Time Division Multiple Access (TDMA) or Frequency Division Multiple Access (FDMA). Due to its relative simplicity TDMA was selected as the initial method to be used in HW testbed.

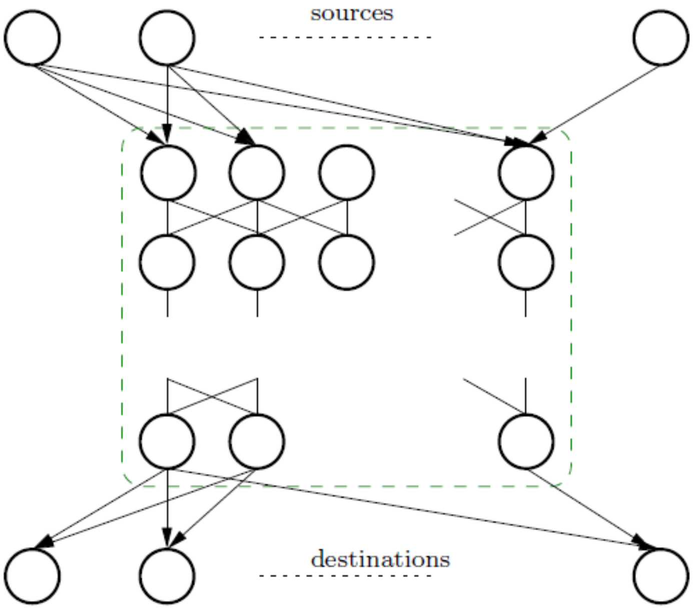


Figure 20: Unidirectional grid cloud scenario (S-RC-a)

The second option, which is more advanced, is a multi-user detection scheme based on successive decoding and interference cancellation techniques. The achievable rate region of this strategy is given by the cut-set-bound theorem. Since most of the points of the rate region are achievable only by time-sharing there is a strong need for cooperation amongst the gateway relay nodes that may incur significant overhead. If the inter-relay communication is complicated or even impossible, e.g. due to great distance among the gateway nodes, the relay output rate should be lowered to meet the limitations of the MAC. However this has impact on the length of time slots in air interface protocol. This option is thus unlikely to be investigated within the realm of the HW demonstrator but could be implemented in the System-Level Simulator (SLS).

2.2.7 Time of flight and asynchronism

Since the SMN testbed platform will demonstrate short range indoor scenarios, time of flight is not an important issue. If the testbed platform were to be extended to large scale topologies then time of flight (and especially different times of flight from different nodes) could become a problem causing mainly time misalignment in the superposition of the different signals. The impact of this time asynchronism on the system BER was studied in the case of simple network with single carrier modulations in [5].

Figure 21 and Figure 22 show how the BER is degraded for various levels of asynchronism for BPSK and QPSK respectively. The case $\Delta t_A = \Delta t_B$ represents a fully synchronised system; the other options introduce some level of time asynchronism. All values are expressed as fractions of the symbol period T_S. As a result it can be stated that a time difference up to t_i = 0.1 T_S does not have a significant impact on the BER. On the other hand, delays higher than 0.4 T_S are catastrophic. QPSK seems to be more sensitive to time accuracy than BPSK.

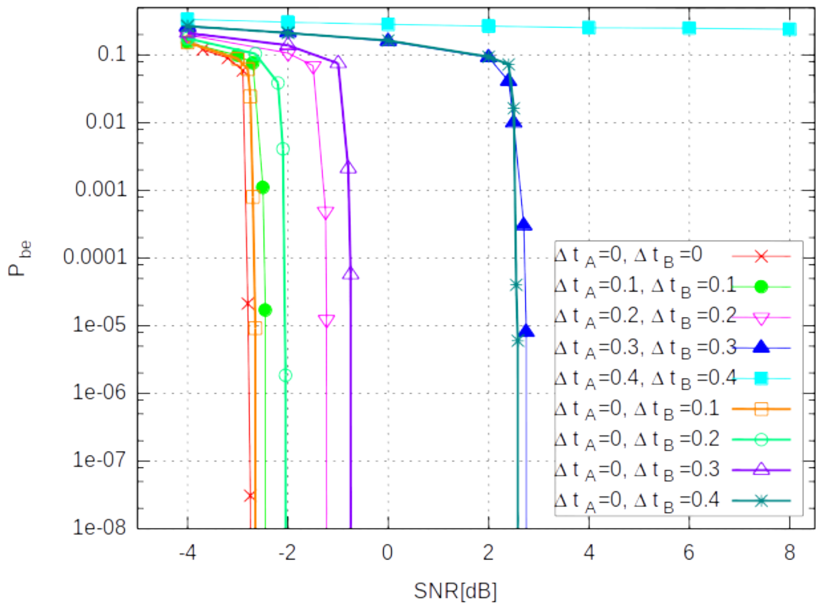


Figure 21: BER for different timing inaccuracies (BPSK) [5]

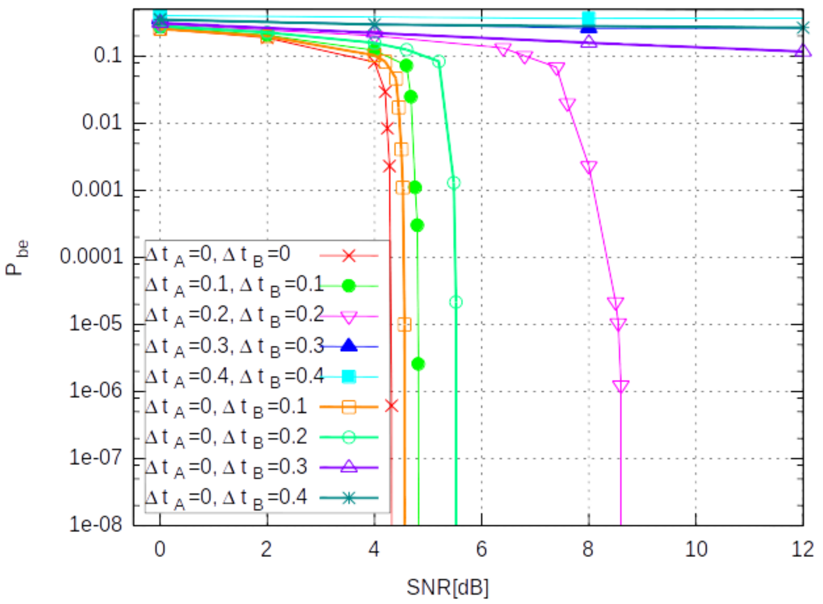


Figure 22: BER for different timing inaccuracies (QPSK) [5]

In [9] they propose an optimal decoding algorithm for time-domain WPLNC with fractional delay, where timing offset is smaller than a symbol period. The algorithm shows that in this setting, symbol misalignment can make the system *more* robust to phase asynchronism. The impact on fractional timing misalignment using practical pulse shaping waveforms is also investigated in [10]. Where the timing offset is more than a symbol period in the time-domain setting, WPLNC has to be divided into two parts: the overlapping and the non-overlapping parts. This is addressed in [6]. When OFDM is used, however, it can be exploited in WPLNC to combat the time mismatch, where the delay in the time-domain is transferred to a phase shift in the frequency domain as long as it is within the cyclic prefix [11]. That is because if the relative symbol delay is within the length of the CP, then the time-domain misaligned samples will become aligned in the frequency domain after FFT [12].

If the time of flight means a serious issue in possible future wide range implementations, it could be solved by classical techniques such as Timing Advance (TA) used in Global System for Mobile Communications (GSM) systems [13]. In GSM systems the TA value is normally between 0 and 63. Each of these steps represents an advance of one bit period (approximately 3.69 μs), which is a change in round-trip distance of about 1100 m. This means that the TA value changes for each 550 m change in the range between a mobile and the base station. It also places a limit of 63 m × 550 m, i.e. 35 km as the maximum cell size. These numbers reaffirm the lack of importance of time of flight correction in short range indoor scenarios.

2.2.8 System under test

The current implementation is a 1-relay butterfly as shown in Figure 23 which supports the BSLD algorithm. In addition a two-source, two-relay, one destination scenario has been implemented (as shown in Figure 11) to implement the DLA. This, however, is only partly 'online' at this stage, as explained in Section 2.2.2.

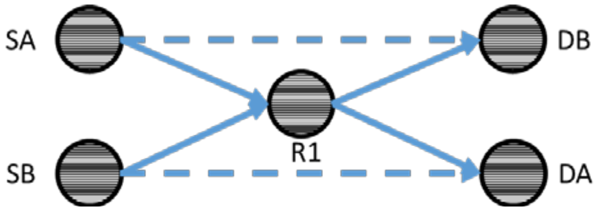


Figure 23: 1-relay butterfly network

Currently this is controlled by four i7 laptops for demonstration purpose: one for the two sources, one for the relay, and one for each of the destinations. For ease of testing in the lab it is possible to run from just three laptops (running the two destinations from one laptop). The next stage is to expand to the 2-relay butterfly (as shown in Figure 4). This will be able to demonstrate both BSLD and DLA fully. This could be run from a minimum of three laptops, although four or even five would be preferable for demonstration purposes.

In order to demonstrate the *full* DIWINE principle where there are terminal nodes, cloud gateway nodes and a cloud node, the hidden node cloud scenario (see Figure 24) would be required. As one laptop is required per stage, this will require at least five laptops and, of course, nine USRPs. This can, however, still all be run through one Gigabit switch as the bandwidth will not become saturated (see Section 2.2.1). A 24 port switch is available for this scenario. If this final target is not achievable within the timeframe of the project, an 8-node implementation, i.e. as Figure 24 without R0 will certainly be achieved. This will be able to demonstrate multiple relay 'stages', a key aspect of the DIWINE paradigm.

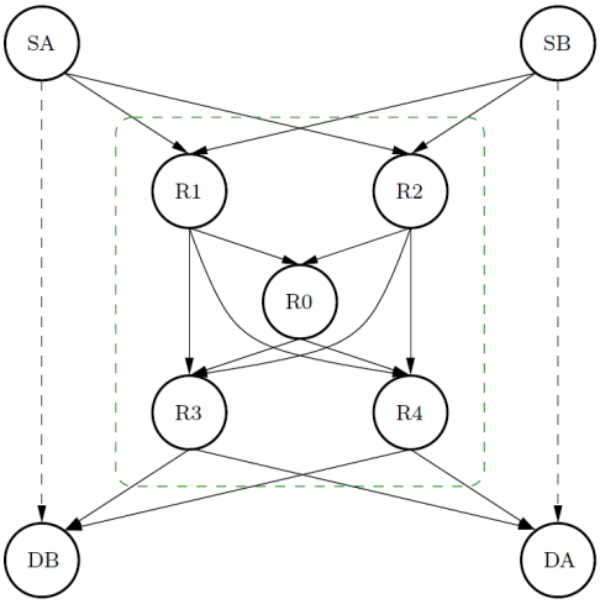


Figure 24: Scenario “Hidden Node Cloud” (S-RC-b)

2.2.9 Node visibility

Since the SMN HW demonstrator will be implemented in indoor scenarios it will experience shadowing caused by obstacles such as walls, furniture etc. This will affect the radio visibility among the nodes and thus define the network topology. The proposed algorithms should cope correctly with this situation. This is the case for DLA (see Section 2.2.2 of [1]) that equips the cloud with the ability to change the WPLNC coding scheme so as to reliably deliver the source information to the destinations. As long as the user nodes (sources and destinations) are connected to at least one of the cloud relays, the cloud will be able to modify itself to provide its service correctly.

None of the proposed WPLNC algorithms needs full connectivity among the nodes, including BSLD and DLA (the two that will be implemented in the SMN demonstrator), neither cloud nor terminal. Initially a static topology of the cloud is assumed, i.e. no nodes are moving so the network topology does not change with time, and no nodes are connecting or disconnecting to/from the network. However the proposed algorithms can be easily modified to support cloud adaptability even for time variable network topology, although there are no plans to implement this in the SMN HW demonstrator.

Some of the algorithms are designed for the simplest cloud consisting of one or two nodes, known as the relay butterfly network (as shown in Figure 25). In this case it is assumed that there is no direct link between the source and its destination, e.g. S_A and D_A or S_B and D_B, in the theoretical work. However this is not the case in the real HW demonstrator deployment. The unwanted direct link is difficult to avoid without also introducing too much error in the wanted direct link which provides the HIS, e.g. S_A to D_B and S_B and D_A). To emulate the theoretical assumption it must be artificially cancelled, using perfect HSI passed through the wired backhaul network using User Datagram Protocol (UDP) source/sink blocks in GNU Radio. This method was already tested whilst implementing BSLD algorithm.

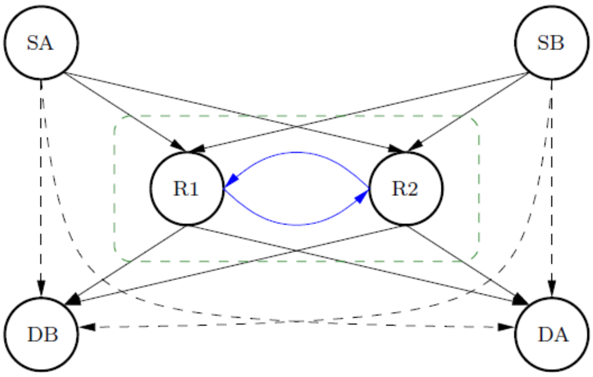


Figure 25: Scenario 2-relay butterfly with perfect intra-cloud coordination (S-RC-b)

However, it should be highlighted that the presence of the direct link is an issue only for the 1-relay butterfly network. It is only in this case that reception of the source transmission directly by the destinations is necessary in order to provide HSI. In larger clouds all the information can be provided by the relays themselves and no HSI from a direct link is needed.

2.3 Distributed network synchronisation related constraints

In this section the hardware limitations affecting the distributed synchronisation algorithm will be discussed, as well as how the SMN testbed has been implemented to take these limitations into account.

2.3.1 Carrier frequency

As detailed in D3.01 [14], the CFO between the nodes causes an apparent frequency rotation in the received data stream. The carrier synchronisation algorithm estimates this CFO and adapts the nodes carrier frequency.

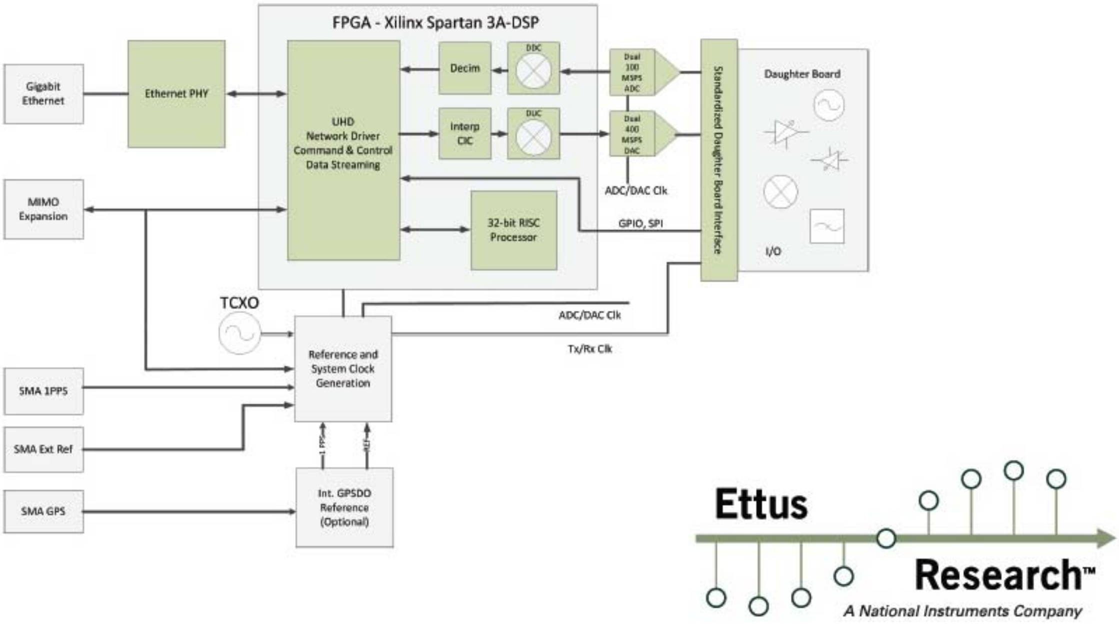


Figure 26: Ettus N210 USRP internal layout [15]

CFO is directly linked to the reference oscillator used within the node, for the USRPs this can either be an inbuilt 100 MHz oscillator, sourced from another USRP via a Multiple Input Multiple Output (MIMO) sync cable, or from an external source. The accuracy and stability of these oscillators will have a direct effect on the CFO. Figure 26 displays the internal configuration of the USRP N210. The block titled 'Reference and System Clock Generation' has multiple inputs illustrating the variety of accepted reference oscillator input sources, Temperature Compensated Crystal Oscillators (TCXO) is the internal oscillator. The output of this block is shown to control the Field Programmable Gate Array (FPGA), digitisers and the Radio Frequency (RF) components of the USRP, as such any variation in the oscillator performance will have a direct effect on these subsystems.

The rest of this section will detail the accuracy and stability of the carrier frequency of the XCVR2450 RF daughter-board, contained within the N210, as well as provide limits based upon worst-case scenarios for the CFO when using different sources of reference oscillator.

2.3.1.1 Frequency Accuracy and Stability

Frequency accuracy is the degree of conformity of a measured or calculated frequency to an ideal or specified reference. The difference is known as the frequency offset, and this changes over time.

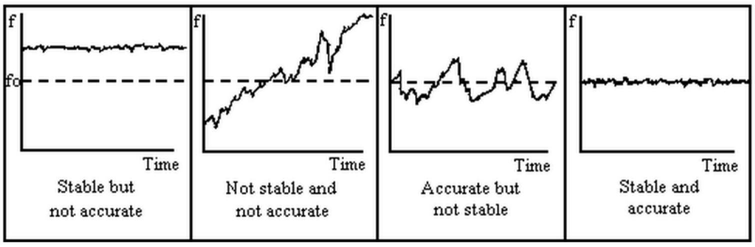


Figure 27: Frequency vs. time

Frequency stability breaks down into three categories:

- Long term stability: measured over hours, days or more;
- Short term stability: measured over periods of fractional seconds to one day;
- Phase noise: deals with very short time scales and produces effects that look like unwanted modulation changing the shape of the waveform rather than a wandering frequency.

2.3.1.2 Phase Noise

Using a spectrum analyser phase noise measurements were made for the XCVR2450 RF board, using both a) no external synchronisation and b) external synchronisation from the Ettus OctoClock-g GPS Disciplined Oscillator (GPSDO) synchronisation distribution system with Global Positioning System (GPS) antenna. It should be noted that in the N210 USRPs all of the RF boards use the master-clock from the motherboard to derive their final LO frequency using Phase Locked-Loop (PLL) synthesis. The master clock rate is 100 MHz and the daughterboard reference clocks are derived from that. Part of the phase noise will be dominated by the master clock, and part by the loop parameters of the PLL. The master oscillators on the N210 are low phase noise TCXOs which have a frequency accuracy of 2.5 ppm, with the external reference GPSDO this falls to 25 parts per billion (ppb), and further to 10 ppb with a lock on the GPS constellation.

RF board	Frequency (GHz)	Offset (kHz)	No sync phase noise (dBc/Hz)	External sync phase noise (dBc/Hz)
XCVR2450	2.48	1	-64	-80
		10	-65	-80
		100	-66	-80
	5	1	-80	-86
		10	-82	-90
		100	-81	-90

Table 5: Phase noise

For the XCVR2450 board it can be seen that the phase noise is high, but it is improved significantly by using external synchronisation. In addition to the phase noise results, it was found in the XCVR2450 that the centre frequency visibly moved over the space of a few seconds, i.e. short term stability is very poor without external synchronisation. This problem was even more noticeable at 5 GHz.

The local oscillator single side band suppression values for the XCVR2450 RF board were similar to those quoted by Ettus. In addition small peaks either side of the sidebands were found which can be attributed to a synthesiser problem, again these were even more noticeable in the 5 GHz band than the 2.4 GHz band.

2.3.1.3 Frequency Offset

In order to assess the frequency offset of the USRPs, a 5 kHz complex sinusoid was transmitted. The transmitter and receiver USRP were connected using a cable. From the baseband waveforms, it was possible to perform sinusoidal analysis assuming that there is only one complex exponential present in the signal. By multiplying the transmitted waveform with the conjugate of the received waveform and estimating the frequency of the resulting signal, it was possible to get an estimate of the frequency offset between the two signals.

The results are shown using the XCVR2450 board in the 2.4 GHz band. In this, and all remaining experiments a sampling frequency of 1 MS/s was used. Results are shown where the receiver and transmitter have a) no synchronisation, b) synchronisation between them by the MIMO cable, c) external synchronisation from the Ettus OctoClock-g GPSDO synchronisation distribution system with GPS antenna. As with the external 10 MHz reference, when using the MIMO cable, the USRPs still use their own oscillator but they should be mutually frequency- and phase-locked, thus the mutual phase noise should be very low. However, there will be a random phase offset every time they are tuned, due to the fractional-*N* synthesis in the synthesiser.

Board	**Centre frequency**	**Frequency offset (Hz)**		
		No synchronisation	**MIMO cable synchronisation**	**External synchronisation**
XCVR2450	2.48 GHz	1105.1	$1.8e^{-3}$	$2.7e^{-4}$

Table 6: Single receiver offsets

It has been noted by Ettus that the high level of integration on the XCVR2450 board is likely to lead to poorer performance compared to their WBX boards. It can be seen that the frequency offset is significantly lower when the MIMO cable is used for synchronisation. As the external reference oscillator has a lower phase-noise and higher-accuracy than the internal clocks, then it is still preferable to use that rather than the MIMO cable, and that is borne out by the results in the XCVR2450 case. It should be noted that rather than using the clock distribution system, it is possible to obtain a GPSDO for each individual USRP, the accuracy of which can further be enhanced using a GPS antenna. Alternatively a third party oscillator such as those produced by Quartzlock may be suitable from improving stability.

The experiment was expanded so that instead of just using a 5 kHz signal, 50 kHz and 500 kHz were also used. It can be seen from Table 7 that the frequency offset is not affected by the frequency of the signal tone. It also shows, though, that there is significant experimental variation in the frequency offset of the USRPs.

Baseband signal frequency	**Frequency offset (Hz)**		
	No synchronisation	**MIMO cable synchronisation**	**External synchronisation**
5 kHz	1105.1	$1.80e^{-3}$	$2.7e^{-4}$
50 kHz	654.7	$1.42e^{-2}$	$5.6e^{-3}$
500 kHz	724.2	$1.00e^{-2}$	$9.2e^{-3}$

Table 7: Single receiver offsets with different signal frequencies

In a second experiment the transmitter USRP is always connected to external synchronisation and transmits at 2.48 GHz with the carrier being modulated by a 5 kHz complex sinusoid. The transmitter is connected to two receivers using a splitter. The receivers have a combination of different synchronisation settings, either a) neither have external synchronisation, b) receiver one has external synchronisation but receiver two does not, c) receiver one has external synchronisation and receiver two is synchronised to this by the MIMO cable, d) receiver one has no external synchronisation and receiver two is synchronised to this by the MIMO cable.

Scenario	Rx1 sync	Rx2 sync	Rx1 frequency (Hz)	Rx2 frequency (Hz)	Difference (Hz)
a)	None	None	7465.7	6845.4	620.3
b)	External	None	4973.6	6962.1	1988.5
c)	External	MIMO cable	4973.6	4973.6	0.035
d)	None	MIMO cable	7706.4	7760.4	0.029

Table 8: Dual receiver offsets

It can be seen that there is a significant difference between the frequency of the received signals in case a) and neither is close to 5 kHz. In case c) the difference is even larger as receiver one achieves very close to 5 kHz with the aid of external synchronisation, but receiver two remains approximately the same as in case a). In case c) It can be seen that the frequency of receiver two is brought very close to receiver one by using the MIMO cable, and both are therefore close to 5 kHz. In the final scenario it can be seen that again the MIMO cable brings the accuracy of the receivers very close to one another, although they are very far from the target 5 kHz.

The following plots show a section of the received waveform from receiver one in case a) and case b). As stated previously, phase noise produces effects that look like unwanted modulation changing the shape of the waveform. In the left plot of Figure 28 very slight modulation can be seen where there is external synchronisation and low phase noise. In the right plot of Figure 28 the modulation effect caused by significant phase noise is much clearer.

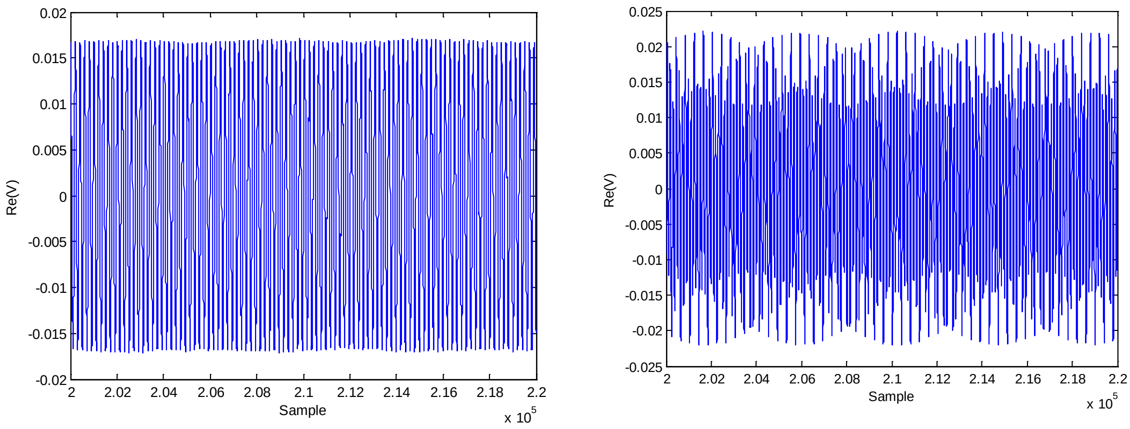


Figure 28: Received waveform;
left: with external synchronisation, right: without external synchronisation

2.3.1.4 Phase offset

It is possible to calculate the phase drifts of the received signals by down-converting with their respective frequencies (as listed in Table 8). The results are shown in Figure 29 below for each of the four scenarios.

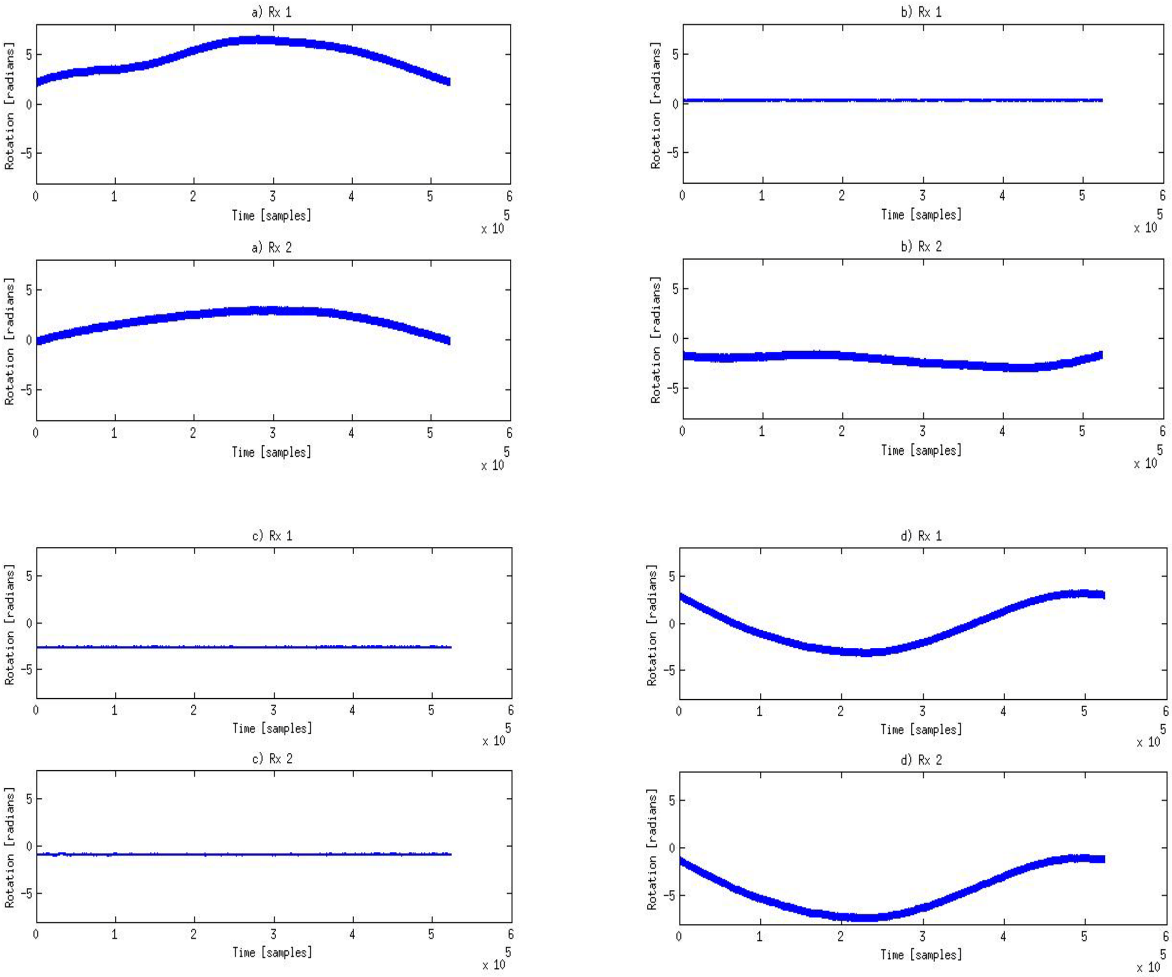


Figure 29: Phase drift of receiver one and two in the different cases: upper-left: case a), upper-right: case b), lower-left: case c), lower-right: case d)

It can be seen that in case a) where there is no external synchronisation that the phase drift is large, and independent between the two receivers. In case b) where receiver one has external synchronisation its phase offset is fixed, but receiver two experiences drift because it has no external synchronisation. When receiver one has external synchronisation, and receiver two is synchronised to it by the MIMO cable, as in case c), they both have a (different) fixed phase offset. In case d), where there is no external synchronisation, but receiver two is synchronised to receiver one by the MIMO cable, there is phase drift. The phase drift is the same in both receivers but with a fixed offset (due to the fractional-*N* synthesis in the synthesiser as discussed earlier).

Figure 30 shows the phase drift with no external synchronisation over 80 samples which is one OFDM symbol using 64 subcarriers and a CP of ¼. It can be seen that there is very significant drift over this duration.

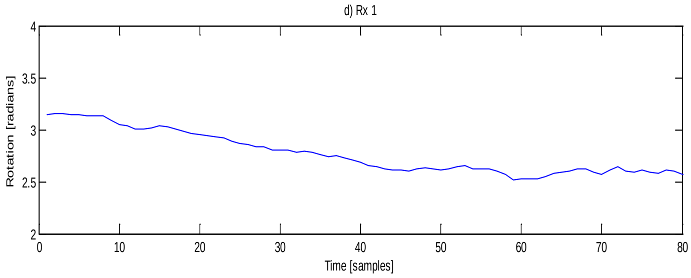


Figure 30: Phase drift of receiver one in case d) over one OFDM symbol

2.3.1.5 Maximum frequency offset

Using the accuracy of the reference oscillators provided in data sheets by Ettus Research [15][16], it is possible to estimate the maximum frequency offset that we are likely to experience when using various oscillators. The oscillators' accuracies are specified in parts per million (ppm) or ppb, from this the accuracy of the reference oscillator can be estimated, then scaled up to estimate the carrier frequency accuracy. Table 9 states the oscillators' accuracy in the XCVR2450 and the OctoClock-g GPSDO with and without a GPS connection. For this work we assume a carrier frequency of 2.45 GHz.

Reference oscillator	**Stated accuracy**	**100 MHz accuracy (Hz)**	**2.45 GHz accuracy (Hz)**
XCVR2450	2.5 ppm	250	6125
OctoClock-g w/o GPS	25 ppb	2.5	61.25
OctoClock-g with GPS	<1 ppb	0.1	2.45

Table 9: Maximum frequency offset caused by various reference oscillators

The data presented in this table shows that if the inbuilt reference oscillators within the XCVR2450 are used for each node, there will be a maximum frequency difference between any two nodes of 12.25 kHz, and this is slightly less than the bandwidth of one OFDM subcarrier, which will be used in the forward and backward data frames. As the synchronisation algorithm is capable of correcting much larger CFO than 6.125 kHz it will be necessary to set a larger artificial CFO at each node, to demonstrate the capability of the integer aspect of the distributed synchronisation algorithm.

2.3.2 Timing offsets

One of the purposes of the synchronisation algorithm is to predict and correct the timing offset (TO) between all nodes. This allows the nodes to have a global time stamp, i.e. all nodes will be able to schedule their transmissions and receptions with respect to other nodes. This concept is discussed in more detail in Sections 2.3.3 and Section 2.4, as the half-duplex constraint complicates the definition of a single global time stamp.

To improve the estimation accuracy of the CFO and TO the symbol rate of the synchronisation frame has been increased from the 1 MS/s of the rest of the system. This allows far greater accuracy in the convergence of CFO and TO within each nodes. To achieve this the signal is sent though either a linear interpolator or matched raised root cosine filter at the transmitter, and processed accordingly at the receiver. This allows finer TO control and better estimation of the CFO. Hardware limitations, such as the Ethernet connection between the USRP and controlling laptop, restrict the interpolation factor to four, resulting in a final sample rate of 4 MS/s.

2.3.3 Scheduling and half-duplex constraints

The essential idea behind the synchronisation algorithm is that each node transmits a Synchronisation Frame (SF) at a certain time, and listens for all the other nodes transmissions. It then iteratively adapts its own timing of its retransmissions to coincide with the other nodes, this process of adaption continues until all nodes are transmitting the SF at exactly the same time and frequency. This transmission of a 4-node network is demonstrated in Figure 31, also shown are the outputs of the matched correlators within the receiver of node 1 (N1), the green represents the outputs for the correlator matched to the SFs' first CAZAC sequence, and the red is from the correlator matched to the conjugate CAZAC (CZ*). Each of the nodes are linked to their corresponding peaks of the correlators. At the end of adaption all the peaks occur at the same time.

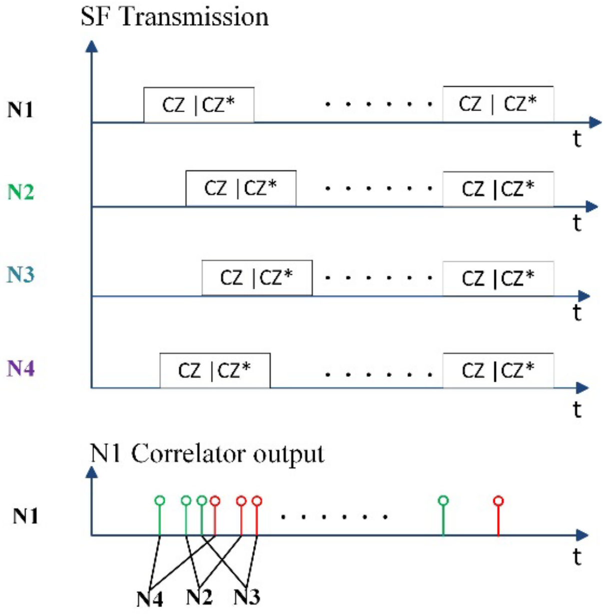


Figure 31: Upper plot: Full-duplex SF transmission and reception,
i lower plot: matched correlator output for N1

The most significant problem affecting the implementation of this system is the half-duplex constraint imposed by the hardware. This results in the nodes not being able to transmit and receive at the same time, and so it would be pointless if all nodes were transmitting instantaneously, as none would be able to receive the transmission. Instead each node has to transmit, and then listen to the other nodes transmissions. This idea is shown in Figure 32.

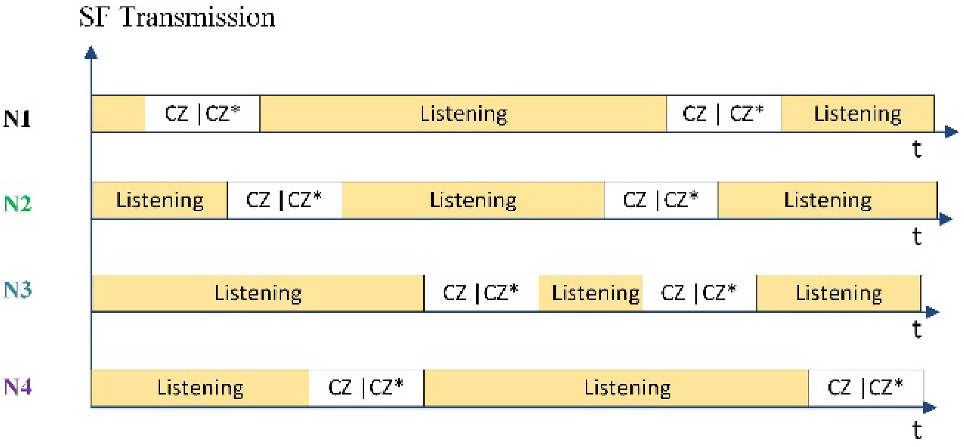


Figure 32: Half-duplex sync frame modification showing SF transmissions and listening periods of each node

An issue that arises with a system like this, is that it is uncertain what reference time the users should adapt to. To overcome this, instead of all the users converging on a single time, they will adapt so their SF transmissions occur at multiples of a certain period. The duration of this period is relatively arbitrary, however for conformity it has been set to be 64 samples long, which is the same as the duration of the CAZAC sequences. The transmissions of a synchronised 4-node network can be seen in Figure 33, displaying how the transmissions of the nodes only occur at set intervals, note the listening periods are now implicit. To implement this in the synchronisation algorithm, instead of taking the timing offset weights to start from the end of the SF transmission, they are now made periodic between these 64 sample boundaries. A further modification is made, so instead of the timing offset weights being set to vary between 0 and 63, as if a SF arrived just one sample too early its offset weight would go to 63, thus creating a large error, when in reality it was only one sample away from being correct. Instead, the timing offset weights vary between -31 and 32, with zero centred on the ideal sampling time. This results in a SF arriving one sample early, receiving a timing offset weight of just -1.

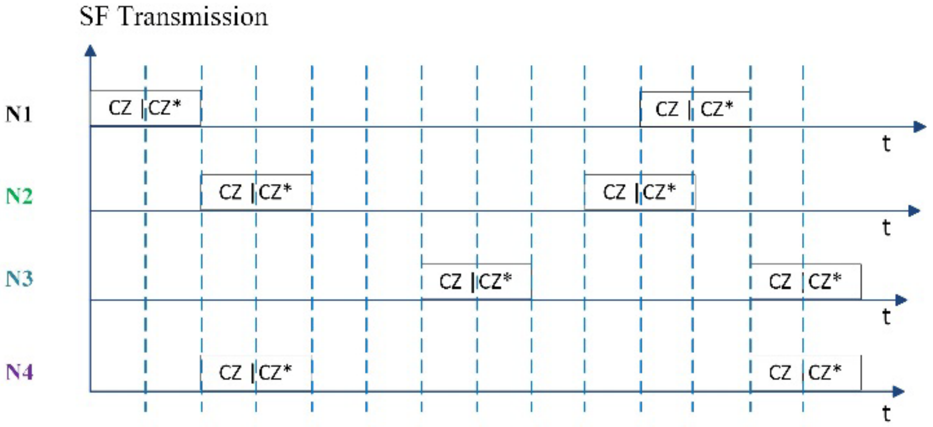


Figure 33: Timing of a synchronised half-duplex system, SF transmissions occur at multiples of a fixed period

A final consideration involved in the processing of the received data is the capture of non-complete SFs. This issue is illustrated in Figure 34, from the perspective of N1, their SF transmissions clash with the first transmission from N2 and the second transmission from N3. This is problematic when the output of the correlators are investigated. It should be remembered that the two matched correlators in each node are matched to separate parts of the SF, so one can still produce a peak even if the other half of the SF is not captured, the missing correlation peaks are illustrated by a dashed line in the middle graph in the figure.

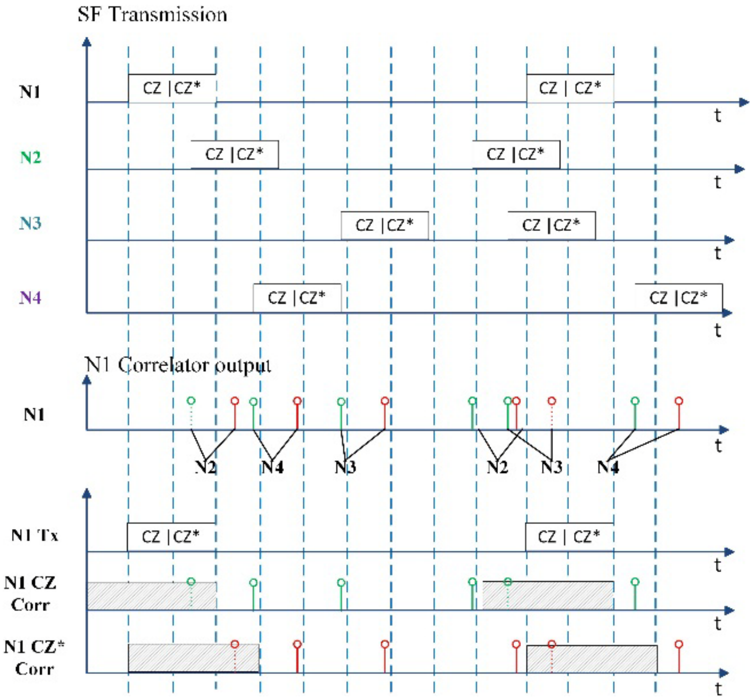


Figure 34: Upper plot: SF transmissions on a 4 node network, middle plot: output of correlators at N1, including showing result of non-complete capture of SF, missing correlation peaks shown using dashed lines, lower plot: modified algorithm, nulling part of the captured data for the two correlators

This non-complete capture of SF at N1 results in one correlator producing more peaks than the other, which in turn can interfere with the CFO and TO estimation process. To avoid this, the captured data is windowed, so that the first correlator, which is matched to CZ, nulls the last 64 samples of the captured data. This is implemented because any correlation peaks produced from this data would correspond to the reception of the second half of the SF that is blocked by N1’s transmission. Likewise, for the correlator that is matched to CZ*, the first 64 samples of data are nulled. This is illustrated in the lowest graph of Figure 34, highlighting the SF transmissions of N1, along with its matched correlator outputs, shading out the regions of the signal that are not analysed by the respective correlators.

A quick note on any SF that are now not fully captured, both correlators will output an equality weighted signal. The peaks produced by the correlators will be proportional to how much of the respective CAZAC sequence has been captured. Figure 35 shows the correlator output of a simulated CZ signal passing a capture window 192 samples long, note how when all the signal is captured the magnitude is unity, then linearly decreases as less of the CZ signal is captured. Also note, capturing less of the signal does not affect the phase of the correlator output, this is important in estimation of the fractional CFO.

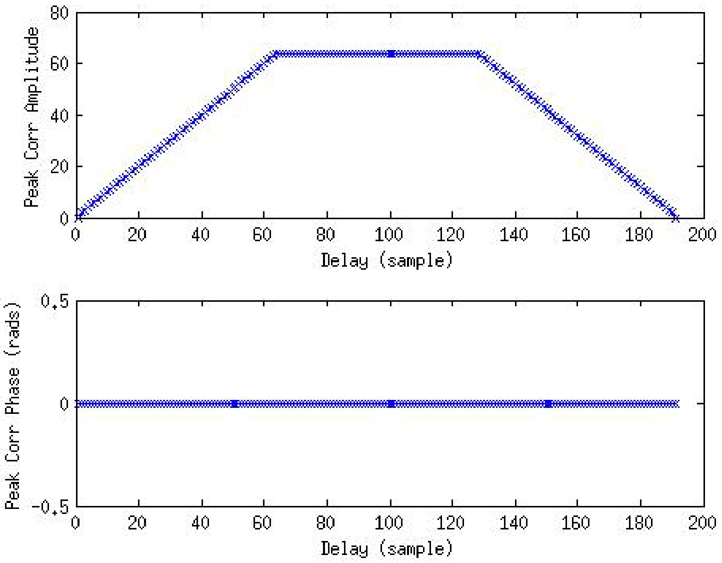


Figure 35: Simulated correlation of partially captured CAZAC sequences

The issue of processing delay also complicates the scheduling of the SF transmissions, this is overcome by inserting another null period just after the capture of the signal, and before the SF transmissions to allow the processing of the received data and adaption of the carrier frequency, this is detailed in Section 2.3.5.

2.3.4 Sampling frequency

Issues with the exact sampling frequency are related to the CFO, TO, and the hardware of the USRP. As shown in Figure 26, the reference oscillator block within the USRP generates signals for the carrier frequency and digitisers. The USRP will use simple multipliers and dividers to attain the correct sampling and RF frequencies. Thus, if several USRPs have been configured to have the same sampling and carrier frequencies, and they are then all connected to a common clock source, they will all have identical frequencies. If, however, they are connected to oscillators that are different, this will result in different sampling and RF frequencies. How much these frequencies differ is all proportional to the difference in the reference oscillators, and so from the difference in the RF signal, the difference in the sampling signal can also be estimated. This is demonstrated in the example below:

If we have two nodes, node A and B, node B is assumed to have a perfect reference oscillator, and so provides a known RF and sampling frequency R_B and S_B respectively. To attain these frequencies, the reference oscillator O_B is multiplied by scaling factors F_{RF} and F_S respectively:

$$\begin{gathered} R_B = O_S F_{RF} \\ S_B = O_S F_S \end{gathered} \tag{6}$$

Equating and rearranging theses to show the scaling factors, which are equal for both nodes, gives:

$$\frac{F_S}{F_{RF}} = \frac{S_B}{R_B} \tag{7}$$

At node A, the carrier frequency is estimated to be α Hz higher than R_B, i.e. $R_A = R_B + \alpha$, the sampling rate of node A can be defined in a similar way $S_A = S_B + \beta$. As the ratio between the reference oscillators scaling factor is the same (7) can be applied to the new equations, giving:

$$\frac{F_S}{F_{RF}} = \frac{S_B + \beta}{R_B + \alpha} \tag{8}$$

Substituting this back into (7) and rearranging for β gives:

$$\beta = \alpha \frac{S_B}{R_B} \tag{9}$$

This shows that the difference in sampling rates is directly proportional to the CFO between node A and B, and the ratio of the sample rate to RF frequency. As a numerical example, Node B has an RF frequency of 2.4 GHz, and a sample rate of 4 MS/s, Node A has a CFO of 6 kHz with respect to Node B. This would give a difference in sample rates between the two systems of 10 Hz. Looking at this in a different way, in 1 second node A will have received 4,000,010 to node B's 4,000,000. This means that node A will receive one more sample for every 400,000 samples that are received at node B.

If this is not processed correctly the TO will drift, and become problematic in both receiving and transmitting. Although, the current transmission system operates in bursts, each burst is generally only a few thousand samples long, so an individual burst may not need modifying on a sample by sample basis. When receiving the signal, and for making global synchronisation possible, the difference in sampling rates would have to be estimated and tracked.

At this point it is worth noting that an optimum method to correct the offset in the carrier frequency would actually be to fine-tune the reference oscillators, this in turn would correct both the sampling and RF offsets, this is the approach proposed in [17]. However the configuration of the N210 USRP does not allow fine-tuning of the reference oscillator.

This presents two possible options for the demonstrator. The first is to implement a system to correct the sampling offset correction, essentially fixing a problem that is very hardware specific and does not have a connection to the concepts of DIWINE. As an implementation of a real DIWINE network could select hardware that allows for the fine-tuning of the reference oscillator, and the CFO correction in the synchronisation algorithm could actually be applied the local oscillator instead, thus fixing both frequency offsets.

The second option is to use an external high-quality reference signal generator, such as the Ettus OctoClock-g GPSDO. Using this the sample timing will very similar on all nodes, even if a different oscillator for each node is used, it will also suffer from far less drift that affects the internal oscillator of the URSP, as discussed in Section 2.3.1.3. A known CFO can be artificially added to each node, this offset could be set far higher than the maximum 6.125 kHz (Table 9) CFO of the USRPs, and would test the convergence of both the integer and fractional CFO estimation of the synchronisation algorithm. A testbed based upon this would demonstrate the concepts of DIWINE, as well as create a more robust testbed, be easier to benchmark performance and be significantly faster to design and debug.

For the reasons stated above, design of the synchronisation testbed will be focused upon the development of the second option, as it will adequately show the concepts of DIWINE, without wasting time on developing a solution to a very hardware specific issue, i.e. N210 USRP. If there is sufficient time, a final system based upon option one may be developed, but not until a fully working testbed based upon option two has been completed.

2.3.5 Processing delay

During the synchronisation phase each node captures a stream of data, this is shown in Figure 33, the CFO and TO are then estimated before a new SF is transmitted. The CFO and TO estimation, and the updating of the USRPs' carrier frequencies take time. To allow for this processing delay the algorithm drops received packets during this estimation and updating phase. Due to the slotted nature of the synchronisation algorithm, as described in Section 2.3.3, the duration between the end of the capture and start of the SF transmissions needs to be a multiple of 64 µs. The final synchronisation algorithm implementation is still being developed, once finished it will be able to estimate the time taken to estimate the CFO/TO and more importantly the time taken to update the USRP. It should be possible to estimate the time taken to calculate the CFO and TO quite reliably, so in the final system will be able to drop packets for a predefined period of time. This delay should be similar in magnitude to what has been discussed in Section 2.2.4. What is not currently know is how quickly the updated can be applied to the USRPs.

2.3.6 Convergence time

The convergence time of the network will heavily depend on the scheduling, and especially the timing of the SF transmissions and the speed these updates are applied to the USRP. Simulations have shown that convergence can be achieved in around 20 iterations, however, due to noise, processing delay and other hardware impairments, the final convergence time will be significantly longer than this. The current implementation is a *master-slave-based* algorithm, as opposed to a *mutually-couple consensus-based* algorithm (see [14] for more details on the distinction). The algorithm is currently achieving convergence in the order of 8 seconds, this is shown for four different CFO values in Figure 36. There are significant optimisations still to be made to this system, as well as implementing the consensus adaption, however it provides an idea of the current convergence performance.

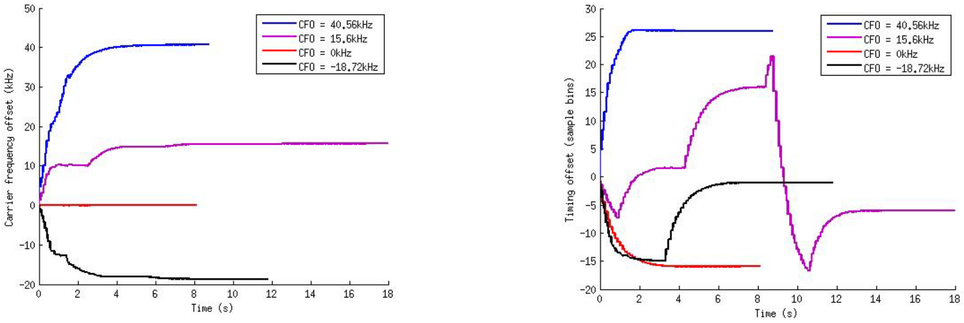


Figure 36: Convergence of CFO (left plot) and TO (right plot) for a master-slave system

The initial system simulations performed by the theoretical partners assumed that all nodes would start transmitting their SF at similar times, separated only by a few hundred samples. However, as they are running at symbol rates in the MHz region, and each node is started by hand, there may be significant differences in the starting of the synchronisation phase of each node, maybe of the order of a few seconds. For this reason it is important that the nodes should not finish their synchronisation procedure too quickly, the current master-slave implementation takes far longer than this, so it is not an issue, however the final system should converge much faster so must take this issue into consideration.

2.3.7 Node visibility

One reason for selecting a consensus-based synchronisation procedure for the final system, rather than a master-slave procedure, was to avoid the issues of hidden nodes. In a large network of nodes it is likely that some nodes would not be able to detect the beacon signal transmitted from a master node, this would result in these hidden nodes not becoming synchronised. The consensus-based approach would not suffer these problems as long as all nodes are connected to the same network, and would of course have the strong advantage of demonstrating the full DIWINE paradigm where there would be no ‘master’ node available.

The physical limitations of the testbed result in the highly likely scenario that the network will be fully connected, and hence a master-slave approach would be sufficient for synchronisation. This allows the two to be benchmarked against each other.

2.3.8 Time of flight

The synchronisation algorithm limits the physical separation between any connected nodes of the network, this distance is proportional to the symbol rate, which for a 1 MHz system is 150 m.

To help explain why this is, take a 3-node network, with nodes N1, N2 and N3 shown in Figure 37, with a symbol rate of 1 MHz. N1 is separated from N2 and N3 by 9 m and 300 m respectively. Giving a time of flight between the nodes N1-N2 of 20 ns and N1-N3 of 1 µs. The network is assumed to be fully synchronised in time and frequency, and all nodes are sampling at the exact same time. At time T1, N1 transmits a SF (for illustration consisting of three samples), this arrives at N2 within a fraction of the symbol duration, and so the TO is estimated to be zero. At N3, the SF now arrives one sample duration after it was expected, thus causing the synchronisation algorithm to adapt its transmissions to be delayed by one transmission. When N3 decides to transmit its SF at time T2, this then arrives two sample durations after it was expected at N1 and N2, causing these two to adapt, moving away from their original synchronised state. For this reason the time of flight must be less than half the symbol duration otherwise the synchronisation algorithm will pull the system out of synchronisation.

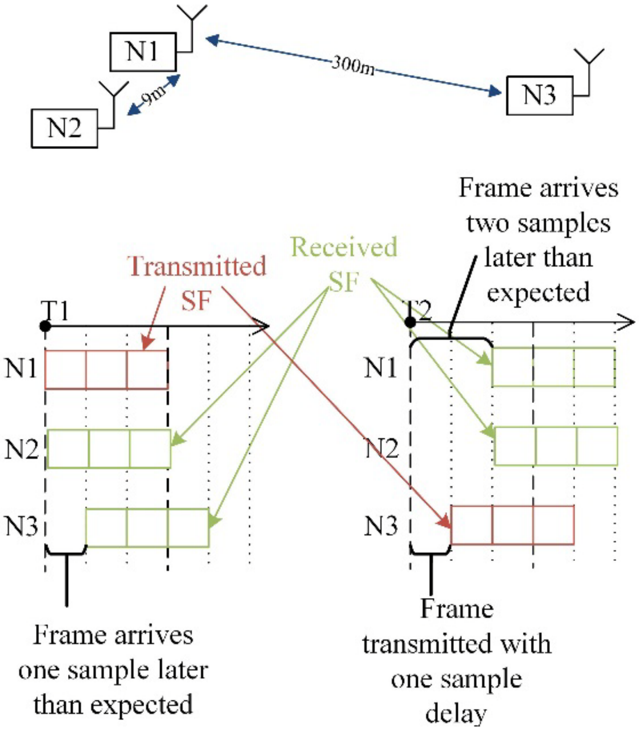

N1
N2
N3
9m
300m
Transmitted SF
Received SF
Frame arrives two samples later than expected
T1
T2
N1
N2
N3
N1
N2
N3
Frame arrives one sample later than expected
Frame transmitted with one sample delay

Figure 37: Three node network, with timing of SF transmissions and receptions

Table 10 shows the maximum separation between nodes for different bandwidth systems.

Bandwidth (Hz)	**Max node separation** (m)
1 k	150 k
500 k	300
1 M	150
20 M	7.5
120 M	1.25

Table 10: Maximum node separation based upon system bandwidth

If there were only two nodes in the network it might be possible to compensate for the longer time of flight by putting a correction factor into the algorithm, however with multiple nodes all transmitting the same SF it is not possible. This will not be an issue for the DIWINE testbed as it all nodes will have a separation of less than 3 m.

2.4 Integration of wireless network coding and distributed network synchronisation algorithms

This report so far has addressed two functional areas of the SMN testbed implementation: WPLNC and distributed network synchronisation. This section goes some way to explaining how the two aspects of the DIWINE paradigm will be integrated.

The initial phase of the system is the cloud initialisation phase, an extension of the Cloud Initialisation Procedure (CIP) reported in [18]. This consists firstly of the consensus-based synchronisation algorithm phase as detailed in Section 2.3.3 and Figure 32, whereby the nodes transmit the SF packets in a half-duplex fashion. This is shown on the left of Figure 38.

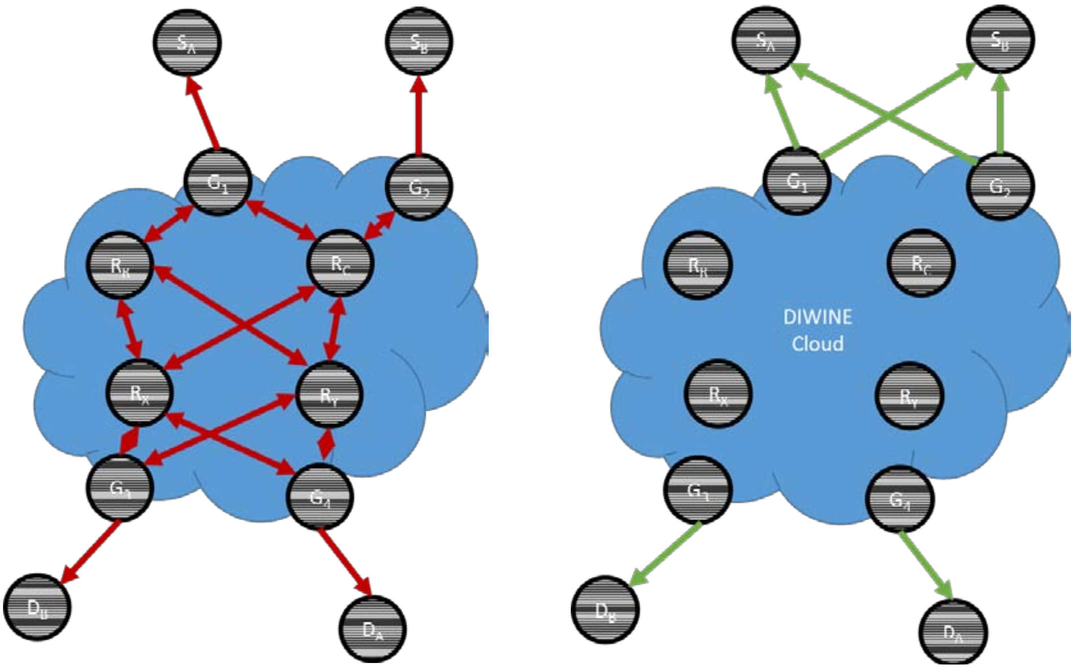


Figure 38: Source, gateway and cloud relay nodes in initial synchronisation phase (left) and transmitting ready packets (right)

Once the CFO and TO of all nodes have converged to a variance within a given threshold, all of the nodes will be time aligned in a slotted fashion as shown in Figure 33. This may take a number of seconds. The cloud and relay nodes then transmit ‘ready’ packets which, in the case of the gateway nodes, are received by the terminal nodes (USRPs) allowing them to synchronise their time registers. This setting of time registers is performed using the UHD command ‘set_time_now’, mimicking the external Pulse Per Second (PPS) signal available in the external synchronisation setup.

The transmission of ready packets is shown in the right of Figure 38. This is crucial to ensure synchronised transmissions, and therefore synchronised superposition of the WPLNC constellations at the relays. These ready packets are two consecutive CAZAC symbols which uniquely identify the node they are transmitted from. In the case that any terminal node receives multiple ‘ready’ packets, the latest to arrive is used as that used to set their initial time. The system then moves to the data phase, leaving the initial synchronisation slot empty, as a back-off to allow time for all nodes to prepare for the data phase. Data transmission can then begin in earnest; the terminal nodes begin transmission in the first forward data frame slot. After the first superframe, the data phase then continues, with interim sync phases at the start of each super frame. The combined air interface is shown in Figure 39.

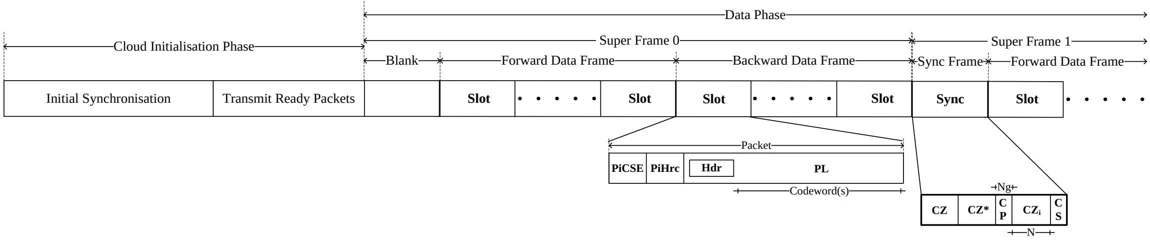


Figure 39: Combined air interface

2.5 Overview of potential advanced prototype limitations

The document so far focuses on the implementation of the SMN based on the use of USRPs. If an application-specific hardware platform were developed (although this is beyond the scope of the current project) some of the limitations would be relaxed and some would be lifted altogether.

With regard to the WPLNC algorithms, the host processing limitations would be removed completely as all processing would take place on the device which may be based on an FPGA implementation. As the device would preferably be low-cost, low-power and have a small form factor, the bandwidth limitations and half-duplex constraint would likely remain in place. The use of OFDM in the ISM band would likely remain the same, as these implementation parameters are not limitations as such, and the desirable DIWINE air interface with DIWINE pilots would also remain. The processing delay using FPGA-based implementation would be reduced significantly allowing real-time operation, and packet lengths could also be extended as these are currently restricted mostly by the GNU Radio implementation. Issues related to multiple access from the gateway to destination nodes would remain unchanged. Using the application-specific implementation would allow more nodes to be implemented so the system under test would grow in scale, which could introduce time of flight issues. This would also affect the node visibility issues; depending on the specific topologies, this could improve or degrade performance.

The use of the USRPs limits the robustness of the distributed synchronisation implementation, as discussed, the inbuilt local oscillator has certain undesirable qualities. First, the frequency drift of the oscillator requires the synchronisation procedure to be run at regular intervals. The second is the inability to tune the reference oscillator, this is problematic for the sampling frequency. Using hardware that has a stable and tuneable clock, all the nodes would be able to operate without any external synchronisation. The absolute accuracy of the clocks would not be important as the synchronisation procedure would correct any absolute frequency offset. Much of the other hardware constraints would remain the same regardless of hardware, as the timing and scheduling are all related to the half-duplex constraint of the SMN, and the maximum size of the network is limited by the system bandwidth.

3 Critical industrial monitoring and control demonstrator

3.1 Introduction

This section of the report provides details about the Critical Industrial Monitoring and Control (CIMC) hardware demonstrator limitations and constraints with respect to the DIWINE industrial scenario. The information provided here will allow tailoring of the DIWINE algorithms within the scope of the demonstrator's HW capabilities, so to minimise integration problems or 'resource-mismatch' issues.

The CIMC hardware demonstrator is based on the integration of a radio module (core processor plus RF transceiver) with a representative WirelessHART industrial sensing node. The radio module was developed starting from advanced state-of-the-art components ensuring a flexible physical layer implementation, a good real-time capability and a very low energy consumption. To allow effective integration with the WirelessHART sensing node, a suitable form-factor, and mechanical interface, were selected.

The CIMC implements a dual-processor, and dual Radio Access Technology (RAT), architecture with one section dedicated to the support of DIWINE-identified algorithms while the other ensures the compatibility with WirelessHART, the current de-facto standard for process-control applications. The two CIMC sections cooperate in the synchronisation and topology management area. The CIMC was carefully designed – and verified – so as to minimise interference between the two RF transceivers entailed by the dual-RAT architecture.

The aim of the CIMC is to allow demonstration of the selected DIWINE algorithms in a context that is in-line with the expected short/medium term industrial scenario. This includes the adoption of packaging solution very near to the final-product one and the adoption of an architecture that takes into account the need for compatibility with the current market standards.

A short overview of the already defined demonstrator hardware and software architecture is first provided. The main CIMC expected limitations and constraints – with respect to the target algorithms and scenarios – is then discussed and quantified. When required, key experimental results are reported and trade-off choices justified in order to provide a more complete picture of the situation.

Finally, some indication is given about which of the limitations and constraints could be relaxed – namely made less severe – in the context of the possible future transition from a 'demonstration' scenario to a 'product' scenario.

3.2 CIMC architecture overview

3.2.1 Layered network architecture

The DIWINE CIMC is based on a layered network architecture, as depicted in Figure 40. Wireless cloud devices are equipped with a dual-RAT operating over the same 2.4 GHz band.

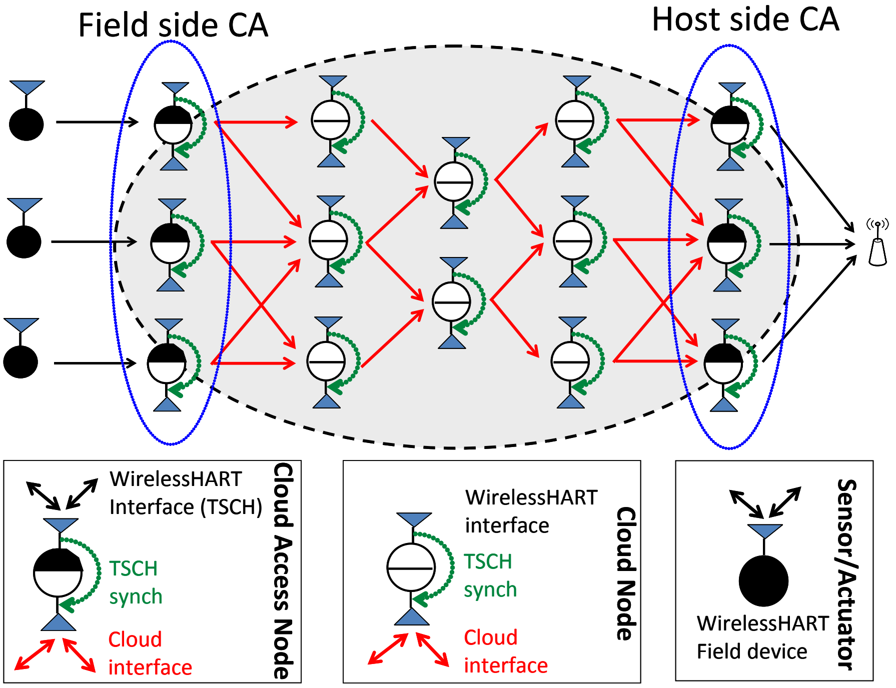


Figure 40: CIMC layered network architecture

The first radio (Time Synchronised Channel Hopping (TSCH), layer 1) supports WirelessHART so as to ensure compatibility with a widely used industrial standard and to allow easier synchronisation and network topology management. The second radio (Cloud interface, layer 2) supports the DIWINE Medium Access Control (MAC) layer as well as the related algorithms [19].

3.2.2 Dual-RAT implementation

The CIMC dual-RAT architecture is implemented as illustrated in Figure 41. As can be seen, two independent 802.15.4 RF transceivers [20] are used, each one associated with a dedicated core processing unit.

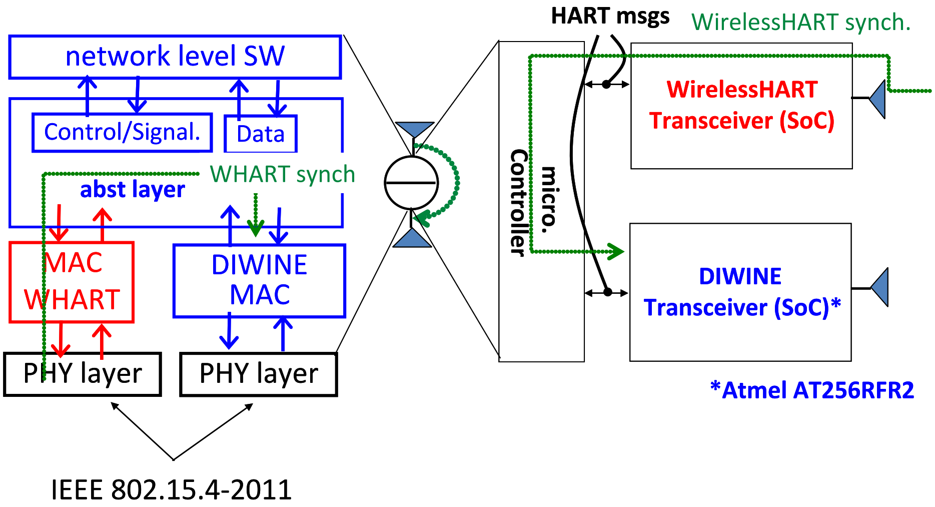


Figure 41: CIMC Dual-RAT architecture

The usage of independent radio modules – with integrated processing capability – was selected both to permit a faster implementation of a demonstration solution and to ensure better compatibility with the existing WirelessHART products. These independent modules are connected via a wired communication channel be based on the HART Transport Layer (TL) format. The WirelessHART and the DIWINE cloud protocol layers will be interconnected so as to exchange 'mesh topology' and 'synchronisation' information.

3.2.3 DIWINE radio module hardware architecture

The DIWINE radio module hardware architecture is depicted in Figure 42.

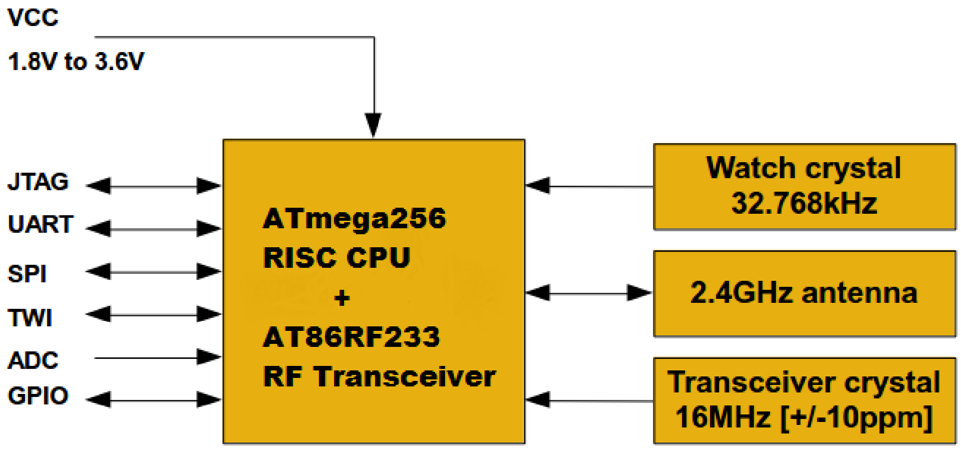


Figure 42: Radio Module hardware architecture

The RF transceiver is based on the AT86RF233 chip, with direct HW-level support for the 802.15.4-2006 standard including the 802.15.4e MAC amendments. Moreover, the AT86RF233 supports a set of advanced features like extended data-rates, better than -100 dBm sensitivity, HW Frame Check Sequence (FCS) computation and advanced low-power modes.

The core processor is based on a 16-Microprocessor without Interlocked Pipeline Stages (MIPS) Reduced Instruction Set Computing (RISC) Central Processing Unit (CPU) (ATmega256) with includes 256 kB of program memory and 32 kB of data memory. The ATmega256 also integrates an HW multiplier, an advanced interrupt handler and many flexible power-save modes with fast wake-up capability.

A 2.4 GHz chip antenna and the high stability transceiver crystal (100 ppm) are also integrated in the module, as well as a low-power watch crystal allowing external wake-up from the low-energy power-down mode. The radio module is based on high integration, System on Chip (SoC) solution, from ATMEL (ATmega2564RFR2 device).

3.2.4 DIWINE radio module software architecture

The DIWINE radio module hardware architecture is depicted in Figure 43.

The 'CLOUD' transceiver will support a simplified broadcast-based MAC layer running on top of a 802.15.4 'extended' physical layer with high data-rate capabilities.

The DIWINE algorithms will be basically allocated at the network/link layer, as no clear-cut network-layer separation is expected to be necessary to fulfil the scope of the CIMC. The application layer will largely be located at the WHART transceiver level, also some application-related configuration will likely need to be integrated at the CLOUD transceiver level.

Finally, an optimised real-time kernel will be used so as to provide a set of services which are as much as possible hardware independent, with a special focus on fast response time, low energy consumption and optimised management of the available power-down modes.

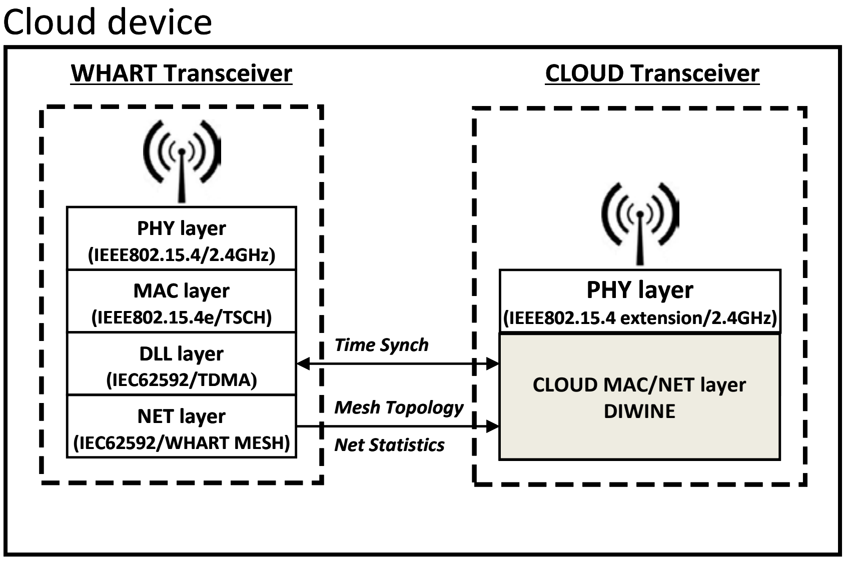


Figure 43: Radio module software architecture

3.3 CIMC limitations and constraints

This section deals with DIWINE radio module limitation and constraints with respect to expected cloud algorithms. In fact, all of the DIWINE CIMC algorithms will be run on the cloud-section radio module. The WirelessHART section will be subject to minor modifications only, which will not impose any relevant constraints or limitations on the applicable DIWINE algorithms.

3.3.1 Processing-power related constraints

3.3.1.1 Peak and average processing throughput.

The DIWINE radio module core processor is an 8-bit, ‘AVR’ family RISC processor ensuring a raw processing throughput of up to 16 MIPS at 16 MHz. It is expected that the core processor will operate at a nominal clock frequency of 8 MHz with the option of switching to 16 MHz when peak processing throughput is required.

Taking into account the overhead related with the operating-system and the MAC layer management, it is expected that it will be possible to allocate a *peak* processing throughput of around 10 MIPS for DIWINE algorithm support.

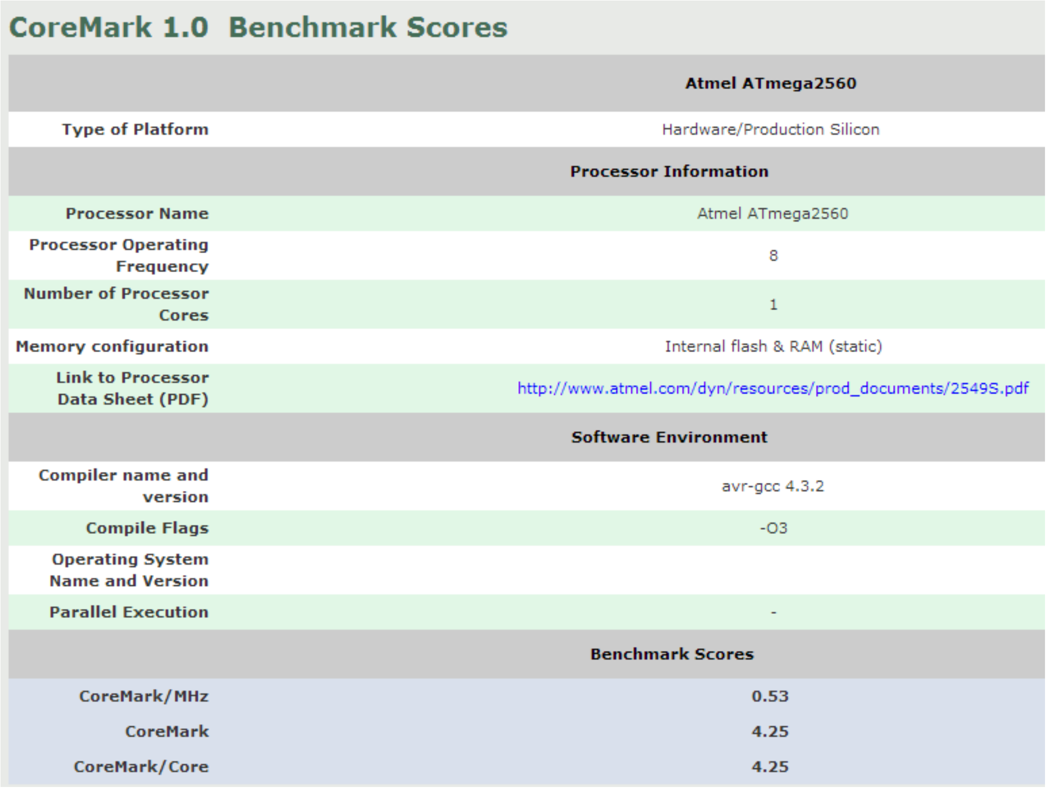
However, due to energy consumption constraints, it is not acceptable for the core processor to constantly operate at a 10 MIPS throughput level for extended periods of time. In fact as the CIMC is not intended to be fully representative of a final product (namely, the final product is likely to be more optimised with regard to energy consumption) it is still necessary to take into account that the adopted algorithms must be reasonably energy-efficient.

Based on these considerations, the *average* processing throughput should never exceed 10% of the peak value, and try to target a figure of 5% as a desirable goal (namely, the limitation is for an average processing throughput between 0.5 MIPS and 1 MIPS). It should however be noted that, due to the typical battery powered nature of the CIMC scenario, also rather long averaging times are viable – namely long ‘bursts’ of peak processing throughput can be accepted.

3.3.1.2 Floating point processing

The DIWINE radio module will not provide any special support for floating-point calculation. In other words, the DIWINE radio module algorithms should be based on *fixed-point* and *integer* calculations only. It is in fact expected that no floating-point calculation will be mandatory for the demonstration of the expected algorithms. Moreover, the integration of a floating-point co-processor within a wireless node compatible with the CIMC industrial scenario is currently not compatible with the expected cost and energy constraints.

3.3.1.3 CoreMark benchmark scores



CoreMark 1.0 Benchmark Scores

	Atmel ATmega2560
Type of Platform	Hardware/Production Silicon
	Processor Information
Processor Name	Atmel ATmega2560
Processor Operating Frequency	8
Number of Processor Cores	1
Memory configuration	Internal flash & RAM (static)
Link to Processor Data Sheet (PDF)	http://www.atmel.com/dyn/resources/prod_documents/2549S.pdf
	Software Environment
Compiler name and version	avr-gcc 4.3.2
Compile Flags	-O3
Operating System Name and Version	
Parallel Execution	-
	Benchmark Scores
CoreMark/MHz	**0.53**
CoreMark	**4.25**
CoreMark/Core	**4.25**

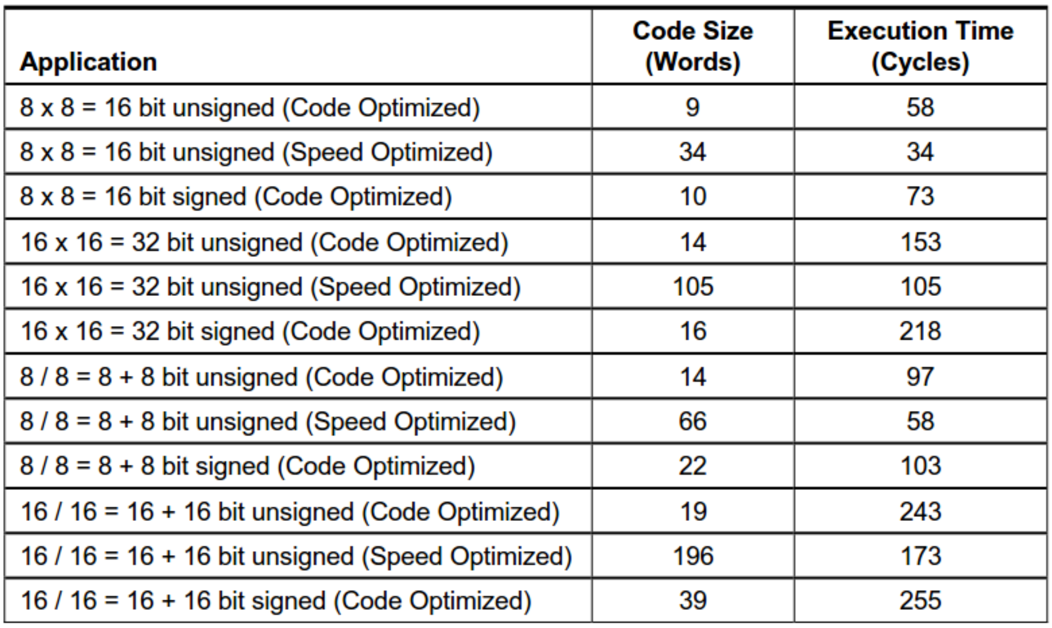
Figure 44: DIWINE radio module CoreMark score

The MIPS score is known to be a rather rough and imprecise way both to assess the processing throughput of a CPU and especially to compare the behaviour of different CPUs making use of heterogeneous instruction sets. To overcome these limitations, more specialised benchmarks are generally used. With reference to embedded core processors similar to the CIMC one, the CoreMark benchmark is often used. The score of the DIWINE radio module core processor with respect to the CoreMark 1.0 suite is detailed in Figure 44.

It could be desirable to run a tentative algorithm on a processor different from the DIWINE radio module one. If this was the case, and if the CoreMark score of the different processor is known, the expected result of algorithm porting could be estimated in advance.

3.3.1.4 Library-based-language integer division/multiplying throughput

The following table, shown in Figure 45, provides a summary of the Software (SW)-based *integer division/multiplication* throughput of the core processor, assuming an optimised assembly-language coding – namely the usage of a library optimised for the specific core (see [25] for more details). Note that the clock cycle time is the reciprocal of the applicable clock frequency (up to 16 MHz maximum).

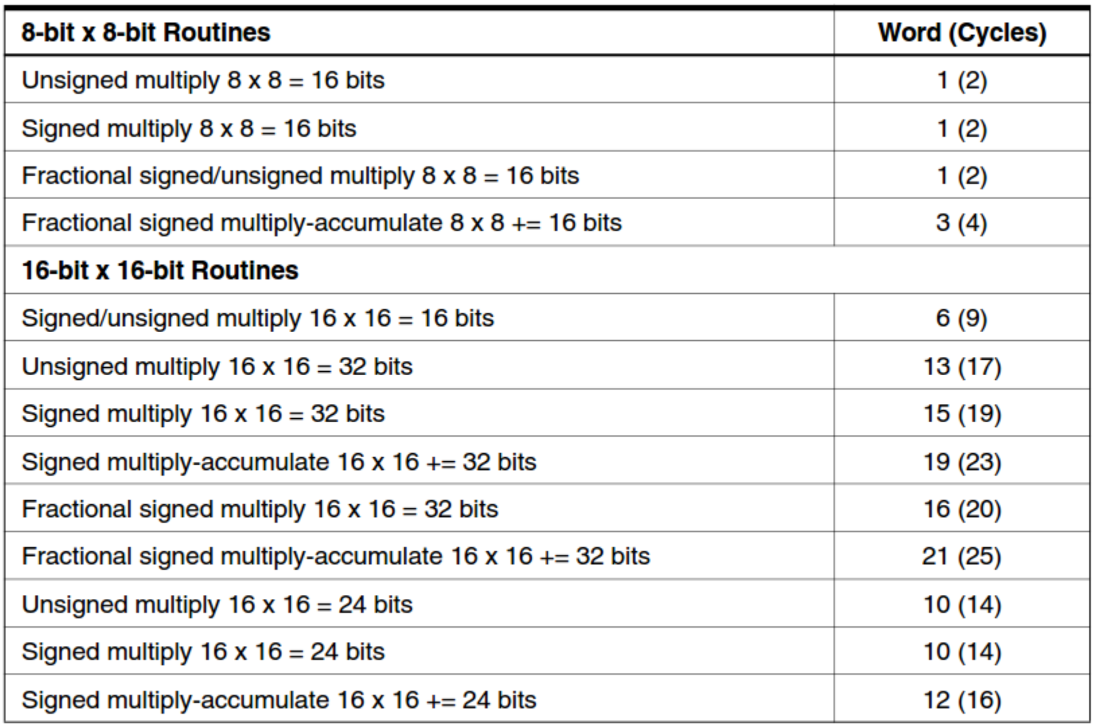


Application	Code Size (Words)	Execution Time (Cycles)
8 x 8 = 16 bit unsigned (Code Optimized)	9	58
8 x 8 = 16 bit unsigned (Speed Optimized)	34	34
8 x 8 = 16 bit signed (Code Optimized)	10	73
16 x 16 = 32 bit unsigned (Code Optimized)	14	153
16 x 16 = 32 bit unsigned (Speed Optimized)	105	105
16 x 16 = 32 bit signed (Code Optimized)	16	218
8 / 8 = 8 + 8 bit unsigned (Code Optimized)	14	97
8 / 8 = 8 + 8 bit unsigned (Speed Optimized)	66	58
8 / 8 = 8 + 8 bit signed (Code Optimized)	22	103
16 / 16 = 16 + 16 bit unsigned (Code Optimized)	19	243
16 / 16 = 16 + 16 bit unsigned (Speed Optimized)	196	173
16 / 16 = 16 + 16 bit signed (Code Optimized)	39	255

Figure 45: DIWINE radio module SW-based integer division/multiplication throughput

The results of the previous table refer to the case when no hardware multiplication support is available. However, in the specific AVR processor version available within the DIWINE radio module does in fact integrate an 8×8-bit hardware multiplier which can complete an operation every two clock cycles.

The following table, shown in Figure 46, provides a summary of the HW-based *integer multiplication* throughput of the core processor, that is to say throughput achievable by means of the integrated hardware multiplier (see [25] and [26] for more details). Also in this case, the usage of a CPU-optimised library is assumed).



8-bit x 8-bit Routines	Word (Cycles)
Unsigned multiply 8 x 8 = 16 bits	1 (2)
Signed multiply 8 x 8 = 16 bits	1 (2)
Fractional signed/unsigned multiply 8 x 8 = 16 bits	1 (2)
Fractional signed multiply-accumulate 8 x 8 += 16 bits	3 (4)
16-bit x 16-bit Routines	
Signed/unsigned multiply 16 x 16 = 16 bits	6 (9)
Unsigned multiply 16 x 16 = 32 bits	13 (17)
Signed multiply 16 x 16 = 32 bits	15 (19)
Signed multiply-accumulate 16 x 16 += 32 bits	19 (23)
Fractional signed multiply 16 x 16 = 32 bits	16 (20)
Fractional signed multiply-accumulate 16 x 16 += 32 bits	21 (25)
Unsigned multiply 16 x 16 = 24 bits	10 (14)
Signed multiply 16 x 16 = 24 bits	10 (14)
Signed multiply-accumulate 16 x 16 += 24 bits	12 (16)

Figure 46: DIWINE radio module HW-based integer multiplication throughput

3.3.1.5 C-language fixed-point division/multiplication throughput

In the following, the *fixed-point* division/multiplication throughput of the core processor is evaluated, assuming an optimised assembly C-coding – namely the usage of C-macros optimised for the specific core. The following code was executed on the core processor; note that this code makes use of the internal hardware multiplier of the core processor.

```
/*! Fixed point math macros. */
#define FIXED_BITS        32
#define FIXED_WBITS       24
#define FIXED_FBITS        8
#define FIXED_TO_INT(a)   ((a) >> FIXED_FBITS)
#define FIXED_FROM_INT(a) (int32_t)((a) << FIXED_FBITS)
#define FIXED_MAKE(a)     (fixed_t)((a*(1 << FIXED_FBITS)))

// Fixed point multiplication routine.
static inline fixed_t fixed_mul(fixed_t a, fixed_t b) {
  return (fixed_t) ((uint32_t) a *  (int32_t) b) >> FIXED_FBITS;
}

// Fixed point division routine.
static inline fixed_t fixed_div(fixed_t a, fixed_t b) {
  return (fixed_t) ((int32_t) a << FIXED_FBIT
```

The following results were obtained:

- Multiplication: 95 cycles;
- Division: 640 cycles.

As can be seen, the fixed-point multiplication throughput is reasonably good. However, the fixed-point division throughput is much worse: in fact the fixed-point division is around 7 times slower with respect to a fixed point multiplication.

3.3.1.6 C-language floating-point division/multiplication throughput

For reference purposes the *floating-point* division/multiplication throughput of the core processor was also evaluated on the core processor. This was done by declaring two C float variables (32-bit IEEE 754 format) and by directly executing the division. The following results were obtained:

- Multiplication: 205 cycles;
- Division: 726 cycles.

As can be seen, the floating-point multiplication throughput is 3.5 times worse with respect to the fixed-point one while the floating-point division throughput is comparable.

3.3.1.7 Division/multiplication throughput summary

Table 11 provides a summary about the division and multiplication throughput, as resulting from previous sections.

	Multiplication (cycles)	**Division** (cycles)
Integer unsigned (library) 16x16=32 16/16=16+16	105	173
Integer 16 unsigned (library, HW support) 16x16=32 16/16=16+16	17	----
Fixed-point signed (optimised C, HW support) 32 (24.8)	95	640
Floating-point signed (optimised C, HW support) 32 (24.8)	205	726

Table 11: Division/multiplication throughput summary

The following conclusions apply:

- Multiplication is always much faster, due to the integrated HW support;
- Fixed-point division is very slow, comparable to floating-point division;
- Assembly language libraries can be used for maximum optimisation.

3.3.1.8 Matrix multiplication throughput

A square ($N \times N$) matrix made of 16-bit integer elements was considered. The time required to multiply two square ($N \times N$) matrices (assuming an 8 MHz core processor clock) was then measured with $N = 8$, $N = 16$ and $N = 32$. The tested multiplication algorithm obtains a square matrix multiplication via N^3 elementary operations, each one made of one 16-bit × 16-bit = 16-bit integer multiplication followed by one 16-bit addition. Much better algorithms are available, but they either require too much memory or become effective only with larger matrices sizes. Using this algorithm, the results in Table 12 were obtained.

Matrix size (*N*)	Elementary operations	Total processing time (µs)	CPU cycles / operation	CPU µs / operation
8	512	9255	72.3	9.04
16	4096	73550	71.8	8.98
32	32768	586983	71.7	8.96

Table 12: Matrix multiplication results

After analysing the adopted algorithm it was observed that some optimisation was still possible. This was basically obtained by bypassing the compiler-generated access to each matrix element (as obtained by pointers manipulation) and substituting it with a more optimised, equivalent in-line code. More details about the original and optimised algorithm are available within Appendix A. Using the optimised algorithm, the results in Table 13 were obtained.

Matrix size (N)	Elementary operations	Total processing time (µs)	CPU cycles / operation	CPU µs / operation
8	512	3510	27.4	3.43
16	4096	27528	26.9	3.36
32	32768	198358	24.2	3.03

Table 13: Optimised matrix multiplication results

As can be seen, the optimised algorithm reduced the processing time to around 33% of the original value. It is expected that if the same algorithm is coded as an optimised assembly library, some further improvement could be obtained.

3.3.1.9 Block data transfer time

It is expected that during the algorithm development some trade-off will have to be considered between allocated data memory and expected processing throughput. The time required for the processor to transfer a block of contiguous data can therefore be considered as a relevant benchmark.

A series of experimental tests on the target core processor performance were therefore performed. A comparatively large data block was transferred and the time required to transfer a single byte was measured – assuming an 8 MHz core processor clock. The results in Table 14 were obtained.

	CPU cycles / transferred byte	CPU µs / transferred byte
C-language data-move algorithm (measured)	14	1.75
Assembly-language data-move algor. (measured)	10	1.25
Loop-unrolling assembly-level limit (estimated)	4	0.5

Table 14: Block data transfer results

It should be noted that the loops unrolling estimations are based on the in-line usage of a series of the following core-processor assembly language instructions – each one requiring 2 cycles to complete as shown in Table 15.

LD	**Rd, Z+**	load indirect and post-inc.	Rd ← (Z), Z ← Z+1
ST	**Z+, Rr**	store indirect and post-inc.	(Z) ← Rr, Z ← Z+1

Table 15: Core processor instructions

Assuming a throughput of 10 cycles x byte, the time required to copy a block of 128 B from one data memory area to another can be calculated as shown in Table 16.

Core processor clock (MHz)	**128-byte transfer time (µs)**
4	320
8	160
16	80

Table 16: Transfer time

3.3.1.10 Data-rate constraints

A number of experiments were performed with the goal of analysing the transceiver capabilities of the DIWINE radio module at the four different data rates available (250 kbit/s, 500 kbit/s, 1 Mbit/s and 2 Mbit/s). The radio modules were deployed in a representative environment as depicted in Figure 47.

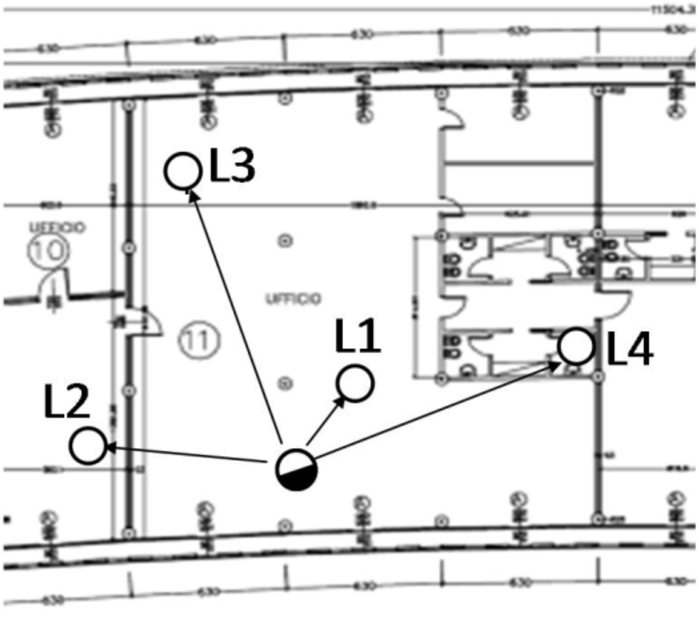


Figure 47: Locations of the DIWINE radio modules for PER evaluation

A series of measurements were performed so as to assess the correlation between the signal strength and the average Packet Error Rate (PER) at the 4 applicable data-rates.

The results of these experiments are summarised in the Figure 48, where L1, L2, L3 and L4 refer to the nodes positions shown in Figure 47. Note that the plot shows the PER as a function of the Received Signal Strength Indicator (RSSI) for each data-rate.

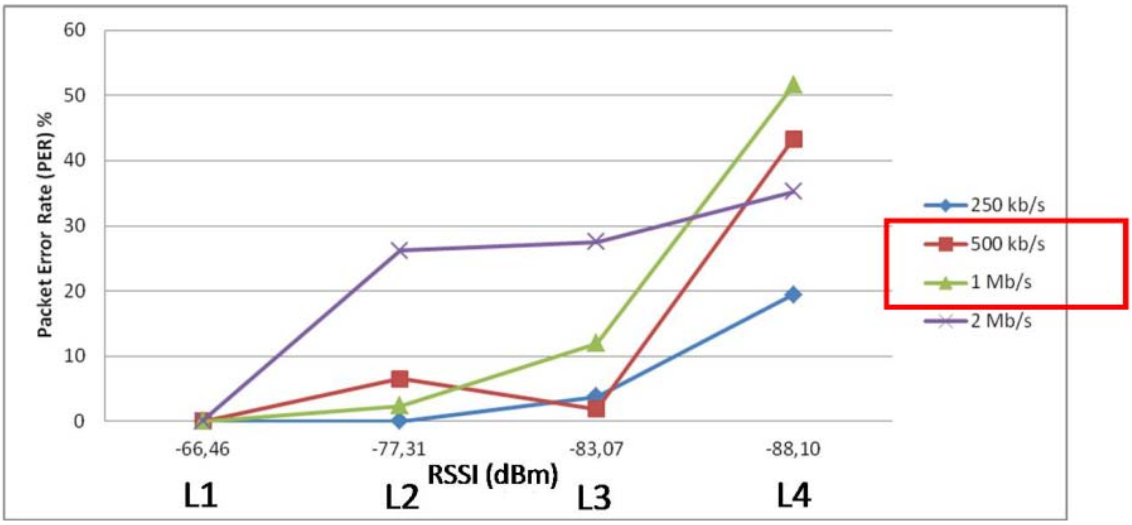


Figure 48: PER versus RSSI for different IEEE 802.15.4 data-rates

From the plot in Figure 48 it can be seen that while the curves for 250 kbit/s and 1 Mbit/s are monotonically increasing as expected, the other two exhibit more unusual behaviour. This is probably due to obstructions and multipath effects that cannot be captured by the RSSI alone.

In summary, a limitation to a data-rate not higher than 1 Mbit/s is to be taken into account, resulting from the need for a reasonable noise tolerance (namely, sensitivity). Moreover, some impairments seem to be evident also when working at 500 kbit/s, in spite of the lower communication speed (further investigation could be required).

3.3.1.11 Throughput constraints

The maximum throughput achievable for the DIWINE radio module at every data-rate was experimentally measured when sending a packet through a two-hop path (a routing path between host and field-side Cloud Access (CA) node, see Figure 49).

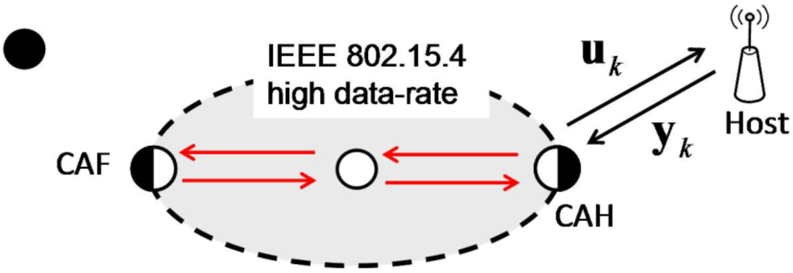


Figure 49: Multihop data-throughput test set-up

The software layer used for the measurement on every node was based on a customised version of the 'demo' 802.15.4 MAC layer made available by ATMEL. This solution is not optimised with respect to the DIWINE scenario, but was selected in order to get some meaningful – though not fully representative – throughput figures in a reasonably short time. The results of the experiments are shown in Figure 50.

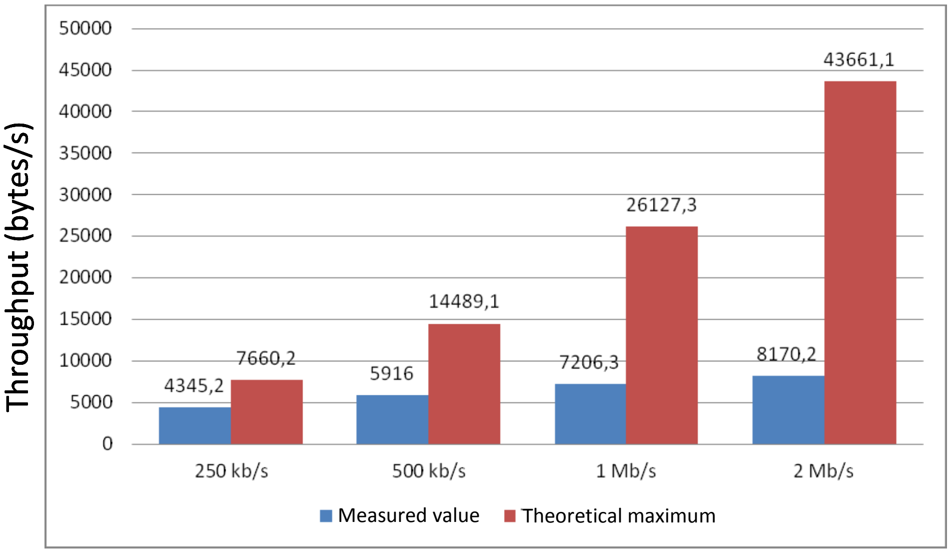


Figure 50: Theoretical maximum vs. measured throughput figures

The diagram shows that, especially at the highest data-rate (1 Mbit/s and 2 Mbit/s), the data-throughput values start to be limited by the processing power capability of the DIWINE module CPU more than by the raw RF data-transfer rate. This should be taken into account in assessing the results obtained by means of the demonstrated algorithms.

However, as noted, these throughput figures refer to a non-optimised MAC layer solution; the final DIWINE MAC implementation is expected to increase the peak throughput at 1 Mbit/s by roughly a factor of 2. The peak available throughput should be limited to around 50% of the theoretical maximum value. In any case, it should be noted that the theoretical maximum value for the throughput assumes a 'zero overhead' condition that would be impossible to achieve also with a fully optimised SW-based MAC. In fact, it could be approached only via a fully HW-based support, which is outside the scope of the CIMC scenario.

3.3.2 Dual-RAT RF constraints

The possible RF-level constraints or impairments resulting from having two parallel-operating radio modules co-located within the same enclosure were investigated. The main problem resulted from possible adjacent-channel interference when one RF module is transmitting and the other is receiving – within the same time-slot. The 802.15.4 standard specifies a rather poor minimum level for the 'adjacent channel rejection' rate (even if real 802.15.4 chips typically exceed the minimum specified level). Therefore, a transmitting 15.4 module could induce some communication errors on a co-located 15.4 module receiving on an adjacent channel.

Experimental verifications were performed on the prototype CIMC implementation so as to detect possible problems of this type. In fact, depending on the relative positions of the two antennas (orientation and distance) and also depending on the selected antenna type (chip antenna or 'sleeve' half dipole) some relevant interference over the receiving module was often detected.

The theoretical estimation of the antenna-coupling effects – to assess the interference level – was too complex to be performed, mainly due to the dominant near-field coupling, which could not be avoided due to the size and mechanical constraints of the selected enclosure. An experimental 'trial-and-error' approach was therefore followed so as to identify the best position for the modules antennas; however, preference was given to the relative orientations ensuring orthogonal polarisation with respect to the radiation pattern of the selected antennas. The results of the experimental measurements are summarised in chart shown in Figure 51. The scenarios are as follows:

- 0 dBm Tx interference over -70 dBm adjacent-channel, Rx – non-optimised antennas position;
- -10 dBm Tx interference over -70 dBm adjacent-channel, Rx – non-optimised antennas position;
- 0 dBm Tx interference over -70 dBm alternate-channel, Rx – non-optimised antennas position;
- 0 dBm Tx interference over -70 dBm adjacent-channel, Rx – optimised antennas position.

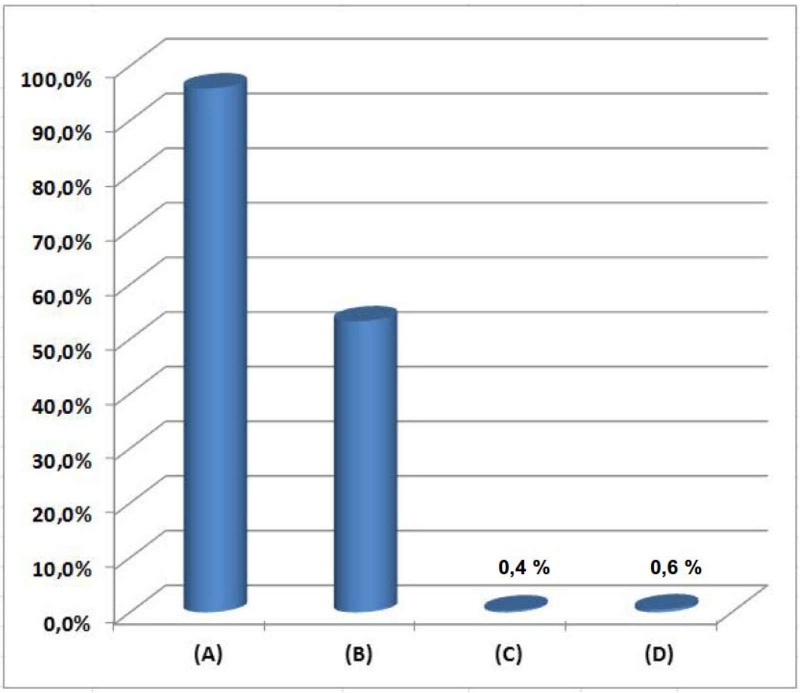


Figure 51: Dual-RAT interference tests (lost messages percentage)

As can be seen, for some antenna layouts/configurations a large number of message drops takes place (A), which persists also when the transmit power is reduced (B) and disappears only when a non-adjacent channel is selected (C). It should also be noted that the improvements from (A) to (C) – at the same Tx level – indicate that the problem is not related with a possible receiver saturation issue, but specifically related to adjacent channel rejection limitations. Finally, an optimised antenna positioning solution was identified (D) where the interference of the Tx module over the Rx module operating on a nearby channel reached a negligible level. This is in fact comparable to the alternate-channel situation (C). The optimised versus non-optimised solution is depicted in the Figure 52.

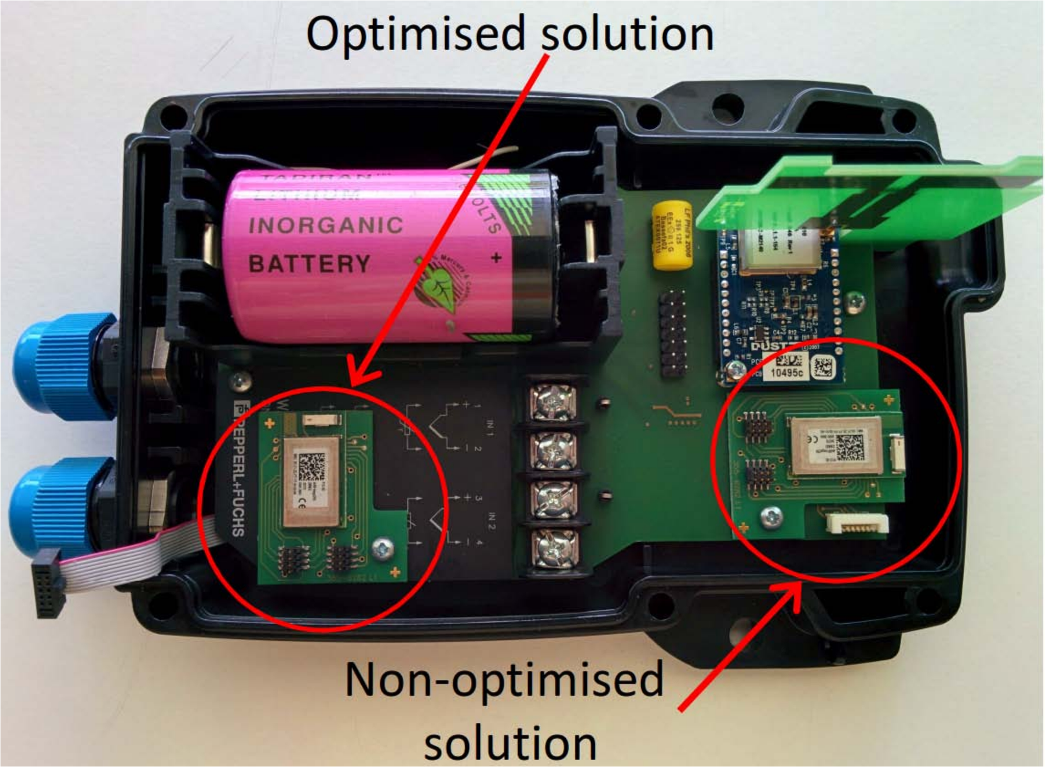


Figure 52: Optimised versus non-optimised DIWINE radio module location.

In conclusion, no special impairments or constraints are to be considered at the RF-level for the algorithm design, e.g. it is not necessary for the Network Layer (NL) to map active slots avoiding adjacent concurrent channels.

3.3.3 Memory resources limitations

3.3.3.1 Program memory constraints

The core processor integrated within the DIWINE radio module makes available 256 kB of Flash program memory. It can be estimated that the real-time kernel code – which is planned to be used to support the upper SW layers – will use 8 kB, while the MAC and physical layer should fit into 32 kB. At the moment we could assign up to 160 kB to the DIWINE algorithms and leave 56 kB spare for contingency uses. Considering the selected scenarios and use cases targeted in D5.51, the DIWINE algorithms are not based on large tables that would require considerable storage. Therefore, it is highly unlikely that there will be any constraints related to program memory limitations.

3.3.3.2 Data memory constraints

The core processor integrated within the DIWINE radio module makes available 32 kB of Random Access Memory (RAM) data memory. In this case, the amount of available data memory is comparative small, and should be allocated carefully. It is expected that all SW elements including and below the MAC layer will us a total amount of data memory equal to 16 kB. This value was derived according to the breakdown in Table 17.

Allocation	Amount (bytes)
802.15.4 frame buffers	1500
MAC payload buffers	2000
Synchronisation histogram	250
Superframe storage area	250
Operating system overhead	2800
Interrupts overhead	300
Other MAC layer overhead	2000
Wired HART buffers	1000
Wired HART statistics and topology	1000
DIWINE statistics and topology	1000
Spare and unplanned	4284

Table 17: RAM allocation

Therefore, half of the total available data memory – namely the remaining 16 kB – will remain available to be allocated for DIWINE algorithm implementation. Considering the comparatively small amount of available data memory – and the difficulty of precise estimations in this area – it is recommended that the amount of allocated memory is carefully monitored (and optimised) during the project evolution, with a clear separation of the ‘algorithm’ memory from the ‘baseline-support’ memory, which should roughly share the available memory resources in a 50/50 way.

3.3.4 Response-time and latency constraints

3.3.4.1 Driver-level latency

The DIWINE module core processor has a nominal interrupt response time of 5 cycles (normal operating mode) and 10 cycles (sleep-mode). For 8 MHz clock cycles this means a latency in the range of 1 µs. While such low latency values could be typically achieved at the physical layer, the algorithms operating above the MAC layer must take into account a much longer latency. This will be shortly discussed in the next section.

3.3.4.2 Application-level latency

Above the MAC layer, the application-level latency must be considered. Namely, no direct access to the HW resources – or to the low-level drivers – will be possible, and this will increase the response time and, more generally, the events service time latency. Considering that a real-time kernel will be integrated within the DIWINE radio module core processor, the application-level latency will be largely defined by the real-time performance of the kernel itself, with reference to the specific processor used within the DIWINE radio module. At this stage of the project, the latency and response time figures shown in Table 18 can be derived from the real-time kernel specifications (no reliable experimental verification was yet possible).

	CPU cycles	**Time (µs) (7.4 MHz clock)**
Context switch time (task-to-task)	346	49.4
Event latency time (timer, I/O, communication)	168	24
Kernel tick time delay (SW timers and pre-emption)	339	2500

Table 18: Latency and response time

3.3.5 MAC-related constraints

3.3.5.1 Synchronisation accuracy limitations

All DIWINE radio modules need to be synchronised – at the MAC-layer level – to properly operate in the expected slotted-access mode. The radio module will adopt an ad-hoc ‘TSCH-assisted’ synchronisation procedure, designed to periodically collect timing information from the WirelessHART radio module and process it so to minimise the worst-case synchronisation error.

The main limiting factor for the MAC layer synchronisation accuracy lies with the jitter that – for a series of reasons – happens to be superimposed on the more precise, on average, timing information. A series of experimental measurements were performed to characterise the jitter distribution over a large number of samples in a representative operating condition. The result of a first set of experimental measurements is presented in Figure 53.

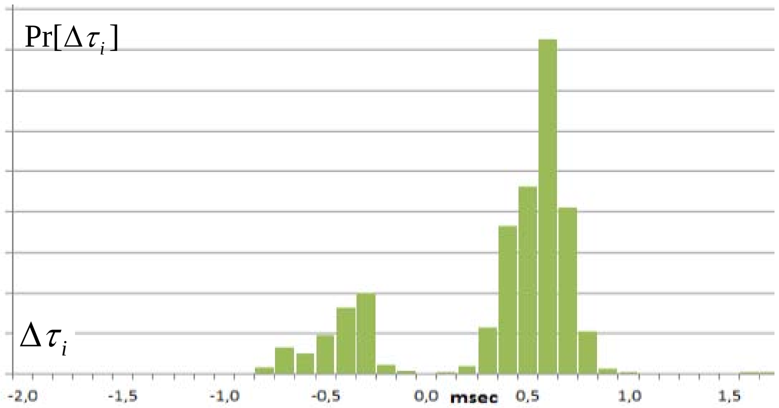


Figure 53: Time jitter distribution – first solution

The Figure 53 diagram indicates that the jitter distribution is around ± 1 ms. A MAC layer synchronisation error of 1 ms was considered an acceptable but not optimal figure, especially at high data-rates where shorter time-slots are typically adopted.

The synchronisation procedure was therefore revised and the communication protocol with the WirelessHART module optimised with the goal of reducing the synchronisation error as much as possible. A second set of experimental measurements was then performed; the corresponding results are presented in Figure 54. The graph indicates that the jitter distribution is around ± 0.4 ms (note that the few elements outside these limits can be easily eliminated. A MAC layer synchronisation of around 0.5 ms can therefore be assumed as the accuracy constraint for proper communication for the DIWINE radio modules.

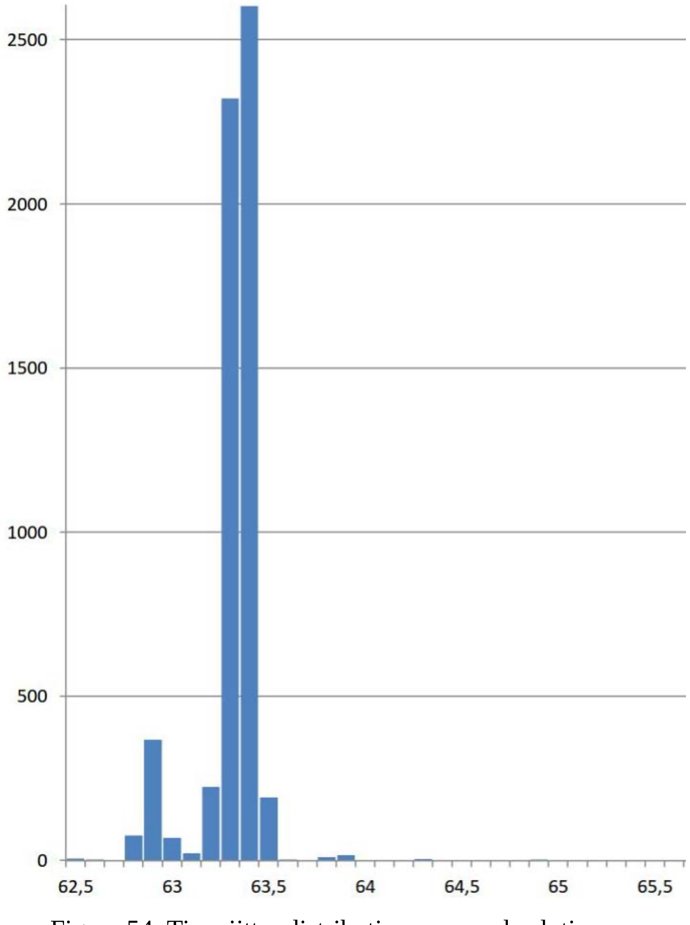
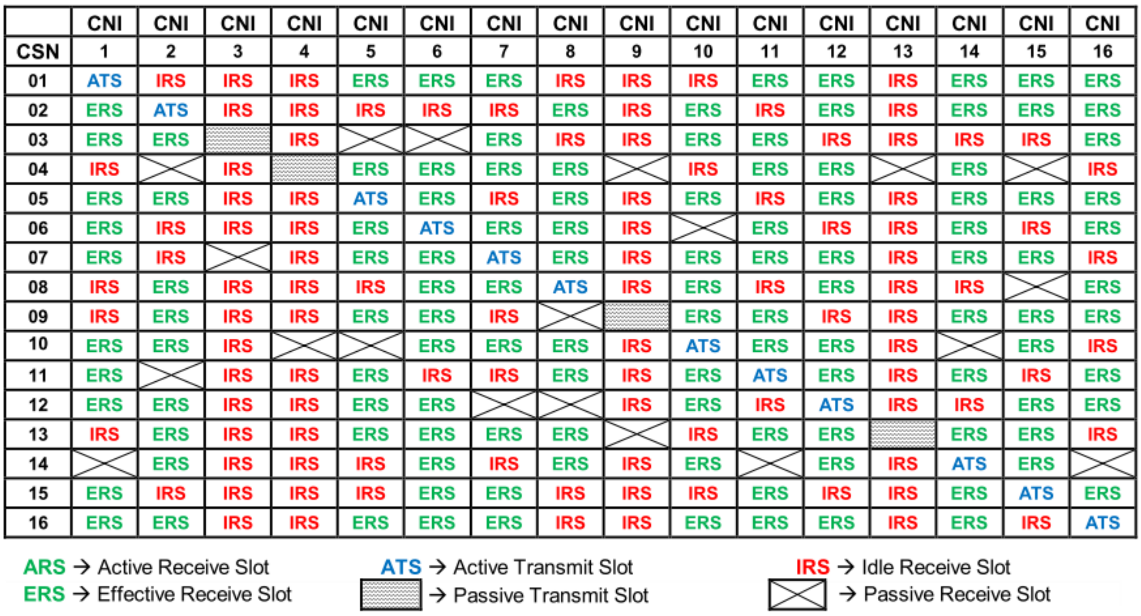


Figure 54: Time jitter distribution – second solution

3.3.5.2 Other MAC layer limitations

The MAC layer solution supported by the DIWINE radio module is based on a baseline superframe as depicted in Figure 55.

According to the current scenario, the MAC-layer will be broadcast-based with integrated support for some limited form of multi-cast and source routing. The DIWINE algorithm will need to take care of implementing, at least in a rudimentary form, the upper-layer capabilities not available at the DIWINE MAC layer level. The adopted baseline superframe structure is defined for 16 wireless nodes. Although different solutions are possible to increase the number of supported devices, this should not be a problem because the CIMC implementation will most likely be limited to 5–10 wireless nodes maximum, due to cost and resources limitations.



	CNI	CNI	CNI	CNI	CNI	CNI	CNI	CNI	CNI	CNI	CNI	CNI	CNI	CNI	CNI	CNI
CSN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
01	ATS	IRS	IRS	IRS	ERS	ERS	ERS	IRS	IRS	IRS	ERS	ERS	IRS	ERS	ERS	ERS
02	ERS	ATS	IRS	IRS	IRS	IRS	IRS	ERS	IRS	ERS	IRS	ERS	IRS	ERS	ERS	ERS
03	ERS	ERS	▨	IRS	☒	☒	ERS	IRS	IRS	ERS	ERS	IRS	IRS	IRS	IRS	ERS
04	IRS	☒	IRS	▨	ERS	ERS	ERS	ERS	☒	IRS	ERS	ERS	☒	ERS	☒	IRS
05	ERS	ERS	IRS	IRS	ATS	ERS	IRS	ERS	IRS	ERS	IRS	ERS	IRS	ERS	ERS	ERS
06	ERS	IRS	IRS	IRS	ERS	ATS	ERS	ERS	IRS	☒	ERS	IRS	IRS	ERS	IRS	ERS
07	ERS	IRS	☒	IRS	ERS	ERS	ATS	ERS	IRS	ERS	ERS	ERS	IRS	ERS	ERS	IRS
08	IRS	ERS	IRS	IRS	IRS	ERS	ERS	ATS	IRS	ERS	IRS	ERS	IRS	IRS	☒	ERS
09	IRS	ERS	IRS	IRS	ERS	ERS	IRS	☒	▨	ERS	ERS	IRS	IRS	ERS	ERS	ERS
10	ERS	ERS	IRS	☒	☒	ERS	ERS	ERS	IRS	ATS	ERS	ERS	IRS	☒	ERS	IRS
11	ERS	☒	IRS	IRS	ERS	IRS	IRS	ERS	IRS	ERS	ATS	ERS	IRS	ERS	IRS	ERS
12	ERS	ERS	IRS	IRS	ERS	ERS	☒	☒	IRS	ERS	IRS	ATS	IRS	IRS	ERS	ERS
13	IRS	ERS	IRS	IRS	ERS	ERS	ERS	ERS	☒	IRS	ERS	ERS	▨	ERS	ERS	IRS
14	☒	ERS	IRS	IRS	IRS	ERS	IRS	ERS	IRS	ERS	☒	ERS	IRS	ATS	ERS	☒
15	ERS	IRS	IRS	IRS	IRS	ERS	ERS	IRS	IRS	IRS	ERS	IRS	IRS	ERS	ATS	ERS
16	ERS	ERS	IRS	IRS	ERS	ERS	ERS	IRS	IRS	ERS	ERS	ERS	IRS	ERS	IRS	ATS

ARS → Active Receive Slot
ERS → Effective Receive Slot
ATS → Active Transmit Slot
▨ → Passive Transmit Slot
IRS → Idle Receive Slot
☒ → Passive Receive Slot

Figure 55: Baseline superframe example at normal network operating mode

3.3.6 Energy-consumption constraints

At this stage of the project, no precise constraint or limitation can be stated about energy consumption issues. It can however be observed that the peak energy consumption is not an issue, while what is more relevant is the long term average energy figures. This results from the fact that the typical CIMC scenario is about battery-powered nodes and does not take into account situations, e.g. energy scavenging, where peak energy consumption could be an issue.

In the CIMC scenario, the availability of 20 Ah batteries can be assumed, and the expected battery life should be between 6 months and 1 year. A real product would likely need a longer battery duration, but the CIMC is not intended to be optimised at this respect, so improvements should be possible. The radio module energy consumption can be subdivided in two areas: core processor energy and RF transceiver energy.

As for the RF transceiver, it should be noted that the energy used in the transmit and the receive mode by is roughly the same. In the CIMC scenario, the ‘on-state’ RF transceiver consumes roughly 13 mA (transmit or receive mode). Therefore, the expected RF-related energy consumption can be computed by estimating the average number of slots where the transceiver is expected to be in the ‘on-state’ (duty cycle). As for the core processor, in the CIMC scenario, an energy consumption of around 5 mA at 8 MHz applies. However, the processor does support many low-power operating modes and the average energy consumption will much depend on the fact that the algorithms are event-driven or throughput driven – and about the associated duty-cycle. As for the core processor, it is also expected that some non-negligible baseline energy consumption will be required when no algorithm is being run, in order to support the core MAC layer operations (synchronisation included). This could be roughly estimated to be around 100–500 µA, but some more precise investigation will be required later in the project so as to confirm or revise these numbers.

3.3.7 Wi-Fi coexistence constraints

This investigation refers to the Wi-Fi 2.4 GHz band. It is envisioned that most ‘process automation’ related wireless technologies will coexist by sharing the same spectrum with other devices employing different radio protocols [21]. Therefore, in view of a practical deployment of the DIWINE demonstrator where heterogeneous radio technologies might coexist over the same spectrum, measurements have been carried out to assess critical interference scenarios and performance limitations focusing in particular on IEEE 802.15.4 Physical Layer (PHY) devices suffering as victims of Wi-Fi interference.

The well-known Link Quality Indicator (LQI) metric for link quality assessment is not enough to estimate the average successful connection probability P_S in the presence of interference [22]. In what follows the problem of coexistence of 2.4 GHz machine type communications conforming to IEEE 802.15.4 and Wi-Fi standards is investigated. In particular, the focus is on the relevant case (in terms of QoS) of WirelessHART devices operating over the IEEE 802.15.4 PHY standard suffering as victims of one Wi-Fi mobile devices. The traffic load of the interference μ is modelled as a Bernoulli process with probability $0 \leq P_\mu \leq 1$ accounting for the degree of frame collisions. The Signal-to-Interference Ratio (SIR) given by $\mathrm{SIR}_l = g_l / \mu_l$ serves as an additional metric to LQI for successful connection probability P_S (expressed in terms of frame error rate) assessment:

$$P_S = \begin{cases} P_r = [g_l > \beta], \qquad if \;\; \mu_l < \dfrac{\beta}{\beta_I} \\ P_\mu P_r[SIR_l > \beta_I] + \left(1 - P_\mu\right) P_r[g_l > \beta], if \;\; \mu_l \geq \dfrac{\beta}{\beta_I} \end{cases} \tag{10}$$

where the model for LQI g_l is based on [15]. The ratio β/β_I indicates the critical value of co-channel disturbance μ (captured by the receiver) above which interference has a relevant impact on connectivity. The Received Signal Strength (RSS) threshold is found as $\beta = -85$ dBm and depends on receiver sensitivity and limits the performance in interference-free scenarios. Sensitivity β also depends on IEEE 802.15.4 PHY data rate settings as described in [22].

The threshold $\beta_I = \beta_I\ (\eta)$ (or link margin) critically depends on the degree of spectrum overlapping $\eta \in [0,1]$ between the useful signal and the co-channel Wi-Fi disturbance. Overlapping is defined as the amount of interference power $\eta \times \mu_l$ lying over the considered IEEE 802.15.4 channel: this is obtained by considering only the portion of Wi-Fi spectrum in common with the IEEE 802.15.4 signal. In what follows, the threshold $\beta_I(\eta)$ is evaluated experimentally for the relevant case (in the industrial context) of IEEE 802.15.4 devices acting as victims of Wi-Fi IEEE 802.11g interference and subject to full ($\eta \geq 0.5$) or partial ($\eta < 0.5$) spectrum overlapping.

A testbed is set up to evaluate the IEEE 802.15.4 robustness through the proposed radio platform. The experiments are conducted using two RF attenuators, a combiner, IEEE 802.15.4 and IEEE 802.11 compliant radio transceivers [23]. Also a spectrum analyser is used to measure the power and losses of the devices in different sectors of the experiment (output attenuators, combiner, crosstalk, etc.). First, the link quality between 802.15.4 radios are tested for a fixed output power and without interference. In the following step, an 802.11 interferer is introduced and its power output is changed in 1 dB steps to identify the first packet loss measured by the 802.15.4 radio devices. Thus, following this procedure it is possible to identify the critical level of 802.11 interference power that cause 802.15.4 packet corruption.

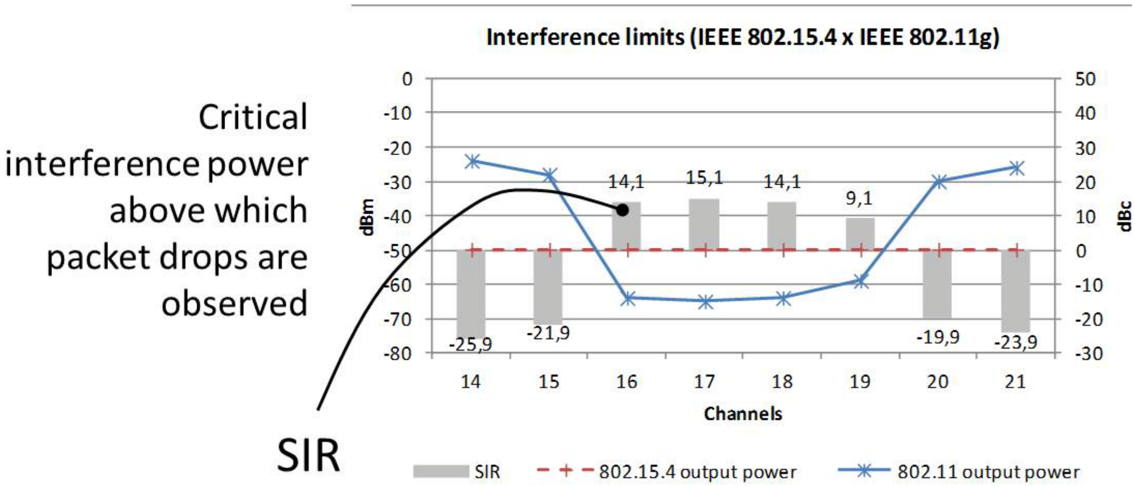


Figure 56: Measurements for IEEE 802.15.4 and IEEE 802.11 spectrum sharing; critical interference power and SIR

Figure 56 shows the minimum values of interference power that cause message losses measured at the 802.15.4 radio receiver. The employed Wi-Fi network is characterised by a high utilisation factor while the IEEE 802.15.4 transmitter continuously sends packets of 127 bytes to its peer receiver.

Results demonstrated that for the 802.15.4 channels (16-19) overlapping with the 802.11 channel 6 the minimum tolerable SIR has values of 14.1 dB, 15.1 dB, 14.1 dB, and 9.1 dB respectively. Also, adjacent channels (15 and 20) might suffer from Wi-Fi interfering signal when the SIR is lower than -21.9 dB and -19.9 dB respectively. Finally, it is measured the lowest tolerable SIR over the alternate channels (with $\eta < 0.1$) observing values of -25.9 dB for channel 14 and -23.9 dB for channel 21.

It is important to emphasise that the results may suffer from slight variations depending on the particular radio platform used, especially interference measured in adjacent and alternate channels. The spurious signals that extend the harmonic frequencies may be more or less suppressed for a specific radio transmitter, also this variation can occur between different channels used in the same radio platform.

3.3.7.1 Experiments in an open LOS/NLOS environment and with variable throughput

In this section we consider the same tests carried out in the previous section now by deploying a real network and considering attenuations due to RF propagation in mixed Line-of-Sight (LoS) and non-LOS environments. Critical (or worst-case) high traffic load scenarios are analysed where a Wi-Fi-Direct Peer-to-Peer (P2P) network and an IEEE 802.15.4 network are possibly continuously transmitting. The impact of enhanced IEEE 802.15.4 PHY data-rate transmission mode is also discussed. The setup consists of one IEEE 802.15.4 device that transmits full data frames of 127 bytes towards a Gateway (GW). The GW node can support double radio technology with Wi-Fi-Direct and IEEE 802.15.4. The transmitter is a programmable device configured to switch among 7 consecutive channels having a bandwidth of 5 MHz (with centre frequencies [20] ranging from 2405 MHz to 2435 MHz). It sends data in continuous mode by disabling Carrier Sense Multiple Access (CSMA) to conform with industry standard PHY [22] and implement a Direct-Sequence Spread Spectrum (DSSS) with factor (Q1 = 8), and data-rate of 250 kbit/s. The GW receiver 1 might be affected by a disturbance (co-channel interference) originated by a Wi-Fi enabled portable Android device communicating in P2P mode with GW node 2 through Wi-Fi-Direct (over IEEE 802.11g) using the band 2400–2420 MHz. The considered interference scenarios are characterised by varying powers μ_l, collision probability P_μ and spectrum overlapping η, both measured by a 2.4 GHz spectrum analyser.

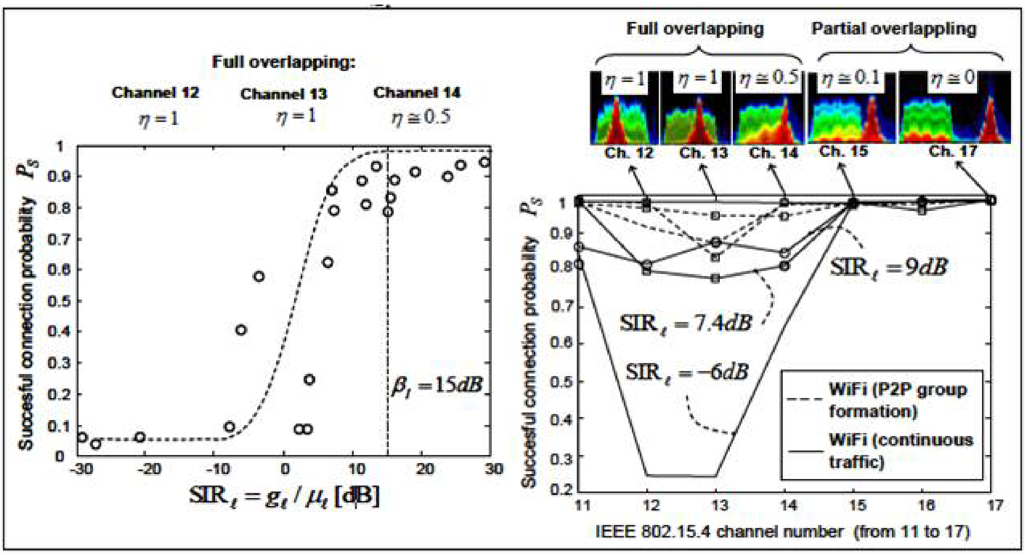


Figure 57: Successful packet transmission probability (IEEE 802.15.4 packets) for varying SIR under full overlapping (left). Successful probability for varying Wi-Fi – IEEE 802.15.4 overlapping (right), for selected values of SIR and traffic loads, under continuous transmission and P2P Wi-Fi group formation

Figure 57Figure 49 shows the analysis of the successful connection probability P_S (10) for varying SIR assuming full overlapping $\eta \geq 0.5$ (channels 11−14) and continuous Wi-Fi traffic, with high load as $P_\mu = 1$. Successful probability is obtained by counting the number of successfully acknowledged data frames normalised by the number of frames received with interferer disabled. According to model (10), the optimal threshold β_I can be reasonably set to $\beta_I = 15$ dB. The use of channels experiencing $\eta \geq 0.5$ must be avoided by blacklisting (when possible) for $\mathrm{SIR}_l < 15\,\mathrm{dB}$. In the same Figure 57, probability P_S is also evaluated over 7 consecutive channels to highlight the impact of spectrum overlapping and interference traffic loads. The analysis focuses on the extreme cases of full overlapping with $\eta = 1$ (channels 11-13) and $\eta = 0.5$ (channel 14), and partial overlapping with $\eta < 0.1$ (channels 15-17), being the most meaningful cases observed in the tests. Wi-Fi Direct P2P group formation is also considered (in dashed lines), with collision probability $P_\mu = 0.1$ and continuous Wi-Fi traffic (in solid lines), with $P_\mu = 1$. The use of partially overlapped channels (15-17) might be reasonably tolerated without significant penalties even at low SIR regime (when $\mathrm{SIR}_l > -6\,\mathrm{dB}$). A reasonable approximation to threshold values (in dB scale) for SIR in (1) is found as

$$\beta_I(\eta, Q_1 = 8) \cong \begin{cases} 15\text{ dB for } \eta \geq 0.5 \text{ ch } (11-14) \\ -6\text{ dB for } \eta < 0.5 \text{ ch } (15-17) \end{cases} \tag{11}$$

and can be used for connectivity prediction.

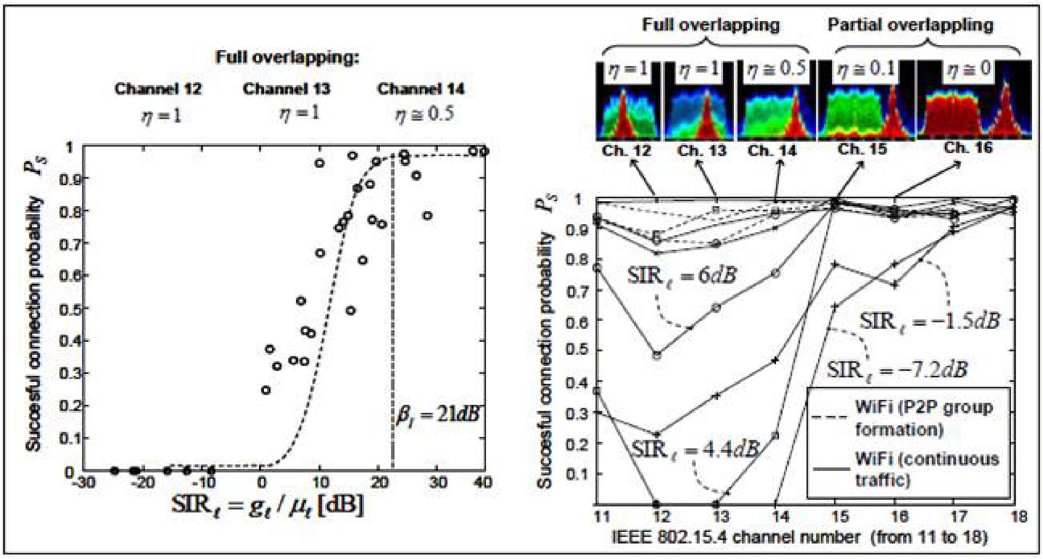


Figure 58: Coexistence results with enhanced IEEE 802.15.4 PHY data-rate devices (1 Mbit/s) with reduced spreading factor to 2. Successful packet transmission probability (IEEE 802.15.4 packets) for varying SIR under full overlapping (left). Successful probability for varying Wi-Fi – IEEE 802.15.4 overlapping (right), for selected values of SIR and traffic loads, under continuous transmission and P2P Wi-Fi group formation

Finally the use of high-data rate mode as required for the DIWINE demonstrator [24]. The use of the enhanced data-rate mode can be a promising option for fast servicing of unexpected conditions that require a fast reaction over the network in a low-latency mode and a meaningful increase of the sensor data publishing rate. In Figure 58 the coexistence with Wi-Fi is addressed for the same settings, now by programming the wireless devices to reduce the IEEE 802.15.4 DSSS factor (for payload transmission) down to a value of $Q_2 = 2$, corresponding to a PHY data rate of 1 Mbit/s. The use of 1 Mbit/s as the data-rate option provides the best trade-off between throughput and reliability (see [24]). According to the experiments, the PHY data frame transmission duration reduces from 4 ms down to 1.6 ms (for a payload of 102 bytes), at the cost of a slightly lower interference-free sensitivity β = -82 dBm, compared to the standard data-rate case. Given that the transceiver is continuously transmitting, the observed collision probability P_μ during P2P Wi-Fi group formation is marginally influenced by the reduced transmission duration (and still $P_\mu = 0.1$). The optimal SIR threshold is now $\beta_I = \beta_I\,(\eta,\ Q_2 = 2) = 21$ dB, and therefore it can be reasonably modelled as a linear function of (11)

$$\beta_I(\eta, Q_2 = 2) \cong \ \beta_I(\eta, Q_1) + \ \left[\frac{Q_1}{Q_2}\right] dB \qquad (12)$$

increasing with the ratio of spreading factors $[Q_1/Q_2]$dB = 6 dB to account for the larger interference capturing effect.

3.3.7.2 Summary of the results

Obtained results highlight tolerable power thresholds for IEEE 802.15.4 devices to coexist with IEEE 802.11 equipment for different levels of interference overlapping. Channels with full overlapping require a minimum SIR equal to 15 dB and co-channel interference with a SIR not higher than -22 dB. In the second setup of experiments in an open (mixed LoS/NLoS) environment the impact of different IEEE 802.15.4 data-rate selections was evaluated. Results show values of threshold SIR in the order of 15 dB and 21 dB for 250 kbit/s and 1 Mbit/s respectively, for 99% successful connection probability in the worst case scenario (considering high data rate and full overlapping channel).

4 Conclusions

This report has evaluated the limitations and constraints of both the SMN hardware demonstrator and the CIMC hardware demonstrator.

4.1 SMN hardware demonstrator

The first half the SMN section looked at limitations relating to the WPLNC algorithms. It was concluded that the current sampling rate of 1 MS/s is sufficient for the implementation of the algorithms, and allows 2 USRPs to be controlled from one laptop. The use of OFDM with 64 bins at a centre frequency in the ISM band at 2.4 GHz is also appropriate. The DIWINE air interface has been updated, including the design and layout of the PiCSE and PiHRC pilots. Testing of the pilots has successfully been undertaken, currently using the existing preamble for CFO and TO correction. Performing channel estimation using PiCSE works well using CAZACs as short as $N_{PiCSE} = 4$, at reasonable SNRs, sufficient to give 0 end-to-end BER in the DLA case.

The half-duplex constraint was reinforced, and both of the algorithms that are to be implemented adhere to this restriction already. The processing delay of the system was shown to be small enough to allow real-time operation with only basic baseband processing, if a number of packets are transmitted in every burst. Once the processing of either the BLSD or DLA is included, however, the processing delay becomes too long and the system runs in a quasi-real-time fashion. This is sufficient to show proof-of-concept of the algorithms. The length of packets is restricted by buffer sizes in GNU Radio, however they are also limited by the necessity for regular synchronisation frames, so the current maximum length of 45 symbols for the PL is more than sufficient.

The multiple access strategy for the final gateway nodes to the destinations will be TDMA, and again the algorithms can support this. The time of flight issue will not affect the system in the indoor testing environment. There should be minimal asynchronism in the system, and using OFDM, any asynchronism less than the CP will result in a phase rotation and will not detrimentally affect the system performance. The system under test has been detailed, where a maximum of a 9-node setup will be implemented – the hidden node scenario. In this case 5 laptops will be required. Node visibility is highlighted as being a minor issue that affects the BSLD where a direct path from source to destination may exist, but this can be resolved by passing the HSI through USRP source/sink blocks in GNU Radio.

The second half of the first section looked at limitations related to the distributed synchronisation algorithm. Hardware impairments of the USRPs were analysed, focusing on the reference oscillators. It was found the internal oscillator suffers from significantly more phase drift and frequency instability compared to Ettus' GPSDO. The benefits of using the GPSDO, as the reference oscillators, with regards to the sampling frequency were also covered, highlighting that the objective of the testbed should be to demonstrate the core DIWINE principals, and to fully test the algorithms developed within the project.

Due to the half-duplex constraint the scheduling of the synchronisation algorithm has been adapted so that all nodes synchronise to multiples of 64 µs, rather than a single global time. Other modifications to improve the CFO and TO estimation were also discussed, such as the increase in symbol rate and windowing the data to cope with non-complete capture of synchronisation frames.

4.2 CIMC hardware demonstrator

Focusing on CIMC hardware demonstrator, this report provided an overview of the main expected limitations and constraints with respect to the target algorithms and scenarios. Key experimental results were reported as instrumental to the adoption of several trade-off solutions for CIMC demonstrator implementation and design. First, processing power constraints of the core processor were highlighted by looking at the energy consumption and power trade-off. Next, according to the envisioned CIMC algorithms illustrated in D5.51, a number of experiments were performed with the goal of analysing the transceiver capabilities and limitations of the DIWINE radio module for different choices of the PHY data rates and dual-RAT configurations. Specific measurement campaigns have been also carried out to assess the expected accuracy of the proposed TSCH-assisted synchronisation scheme. Finally, in view of the possible deployment of the CIMC demonstrator in a practical industrial environment, an ad-hoc analysis of coexistence in unlicensed spectrum sharing scenarios has been carried out. Obtained results highlight the tolerable power thresholds for IEEE 802.15.4 PHY devices to coexist with IEEE 802.11, i.e. Wi-Fi, equipment for different levels of interference overlapping.

Bibliography

[1] David Halls, Pavel Procházka, Tomáš Hynek, Maria Antonieta Alvarez Villanueva, Vittorio Rampa, Jan Sýkora, Umberto Spagnolini, Eduard A. Jorswieck, Kostas Ramantas, David Boxiade, Roger Torne, and Pin-Hsun Lin, "SMN platform description and high-level routines", DIWINE deliverable D5.41, March 2014.

[2] Timothy M. Schmidl, Donald C. Cox, "Robust frequency and timing synchronization for OFDM", *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, December 1997.

[3] Koralia N. Pappi, George K. Karagiannidis, and Robert Schober, "How sensitive is compute-and-forward to channel estimation errors?", Proceedings of the *IEEE International Symposium on Information Theory* (ISIT), 7–12 July 2013.

[4] Keyvan Yasami, Abolfazl Razi, and Ali Abedi, "Analysis of channel estimation error in physical layer network coding", *IEEE Communications Letters*, vol. 15, no. 10, pp. 1029 –1031, October 2011.

[5] Jan Sýkora, Miroslav Hekrdla, Pavel Procházka, Tomáš Uřičář, and Tomáš Hynek, "Network, resource, and interference aware coding and decoding", SAPHYRE deliverable D3.2b, December 2012.

[6] Yixin Li, Fu-Chun Zheng, and Michael J. Fitch, "Physical-layer network coding with channel and delay estimation", *IET Communications*, vol. 7, no. 11, pp. 1109–1116, July 2013.

[7] Jan Sýkora, Tomáš Uřičář, Pavel Procházka, Tomáš Hynek, Eduard A. Jorswieck, Zuleita K. M. Ho, Kostas Ramantas, Orestis Georgiou, Alister G. Burr, Umberto Spagnolini, Monica B. Nicoli, Stefano Savazzi, and Stefano Galimberti, "Definition of scenarios, models, performance metric and utility targets", DIWINE deliverable D2.01, September 2013.

[8] Ettus "UHD Latency",
http://code.ettus.com/redmine/ettus/projects/uhd/wiki/Latency.

[9] Soung-Chang Liew, Shengli Zhang and Lu Lu, "Physical-layer network coding: Tutorial, survey, and beyond", *Elsevier Physical Communication, Special Issue on "Network Coding and its Applications to Wireless Communications"*, vol. 6, pp. 4–42, March 2013.

[10] Yixin Li and Fu-Chun Zheng, "A simple physical-layer network coding with fractional delay", Proceedings of the *IEEE International Conference on Communications in China* (ICCC), 12.–14. August 2013.

[11] Francesco Rossetto and Michele Zorzi, "A practical architecture for OFDM-based on decode-and-forward physical layer network coding", *IEEE Transactions on Signal Processing*, vol. 60, no. 9, pp. 4747–4757, September 2012.

[12] Lu Lu, Taotao Wang, Soung-Chang Liew, and Shengli Zhang, "Implementation of physical-layer network coding", *Elsevier Physical Communication*, February 2012.

[13] 3GPP TS 05.10 “Radio subsystem synchronization”, http://www.3gpp.org/DynaReport/0510.htm, accessed 2014-09-29.

[14] Umberto Spagnolini, Monica B. Nicoli, Gloria Soatti, Alessandra Pascale, Vahid Forutan, Ali Parichehreh, Stefano Savazzi, Jan Sýkora, Tomáš Hynek, Eduard A. Jorswieck, Alister G. Burr, and Mehdi Mortazawi Molu, “Cloud network processing for core demonstration scenarios”, DIWINE deliverable D3.01, September 2013.

[15] https://www.ettus.com/content/files/07495_Ettus_N200-210_DS_Flyer_HR_1.pdf, accessed 2014-09-29.

[16] https://www.ettus.com/content/files/Octoclock_Spec_Sheet.pdf, accessed 2014-09-29.

[17] Omid Abari, Hariharan Rahul, and Dina Katabi, “One clock to rule them all: A primitive for distributed wireless protocols at the physical layer”, 27. April 2014.

[18] Tomáš Hynek, David Halls, and Jan Sýkora, “Practical implementation of cloud initialization procedure for wireless physical layer network coding clouds”, Proceedings of the *European Wireless Conference* (EW), 14.–16. May 2014.

[19] Leonardo Ascorti, Stefano Savazzi, and Stefano Galimberti, “Cloud-based WirelessHART networking for Critical Industrial Monitoring and Control”, Proceedings of the *IEEE International Conference on Industrial Informatics* (INDIN), 27.–30. July 2014.

[20] Standard IEEE 802.15.4-2006, “Part 15.4: Wireless Medium Access Control (MAC) and Physical layer (PHY) specifications for low-rate Wireless Personal Area Networks (LR-WPAN)”, 2006.

[21] Wenqi Guo, William M. Healy, and Zhou Meng-Chu, “Impacts of 2.4 GHz ISM band interference on IEEE 802.15.4 wireless sensor network reliability in buildings”, *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 9, pp. 2533–2544, September 2012.

[22] Stefano Savazzi, Vittorio Rampa, and Umberto Spagnolini, “Wireless cloud networks for the Factory of Things: Connectivity modeling and layout design”, *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 180–195, April 2014.

[23] Jean M. Winter, Ivan Müller, Stefano Savazzi, Leandro Buss Becker, João C. Netto, and Carlos E. Pereira, “Coexistence issues in wireless networks for factory automation”, Proceedings of the *IEEE International Conference on Industrial Informatics* (INDIN), 27.–30. July 2014.

[24] Stefano Galimberti, Stefano Savazzi, Leonardo Ascorti, and David Halls, “CIMC radio module platform and WNC algorithms”, DIWINE deliverable D5.51, March 2014.

[25] ATMEL application note, “AVR200: Multiply and Divide Routines”, Rev. 0936D-AVR-09/09, accessed 2014-09-29.

[26] ATMEL application note, “AVR201: Using the AVR® Hardware Multiplier”, Rev. 1631C–AVR–06/02, accessed 2014-09-29.

5 Appendix A – CIMC detailed estimations

5.1 Matrix multiplication algorithm

5.1.1 Overview

This benchmark measures the time required to multiply two square matrices ($N \times N$) using the naive algorithm that requires N^3 multiplication and the same number of sums (complexity: $O(N^3)$). The asymptotically faster Strassen's algorithm, which is usually implemented in linear algebra libraries, is less suitable for microcontrollers, because of the much larger memory requirement and the low performance improvement in the case of small matrices. The algorithm has been implemented in C in two different versions, as in the following.

5.1.2 Simpler implementation

This is the naïve implementation, a short and very readable piece of code that handles the matrices as two-dimensional arrays:

```
for (uint8_t i = 0; i < MATRIX_SIZE; i++) {
    for (uint8_t j = 0; j < MATRIX_SIZE; j++) {
      for(uint8_t k = 0; k < MATRIX_SIZE; k++) {
        res[i][j] += m1[i][k] * m2[k][j];
      }
    }
  }
```

However, this implementation turned out to be very inefficient: in fact, the compiler needs an extra multiplication and a sum to convert the two dimensional index into a pointer offset and since three matrices are accessed, four 16-bit multiplications are performed at each iteration instead of one. Actually, the time wasted to compute the pointers is more than the one use to perform the matrix operation itself.

5.1.3 Optimised implementation

In order to reduce the time overhead, a different implementation of the same algorithm was developed that avoids multiplications on pointers. The code is longer and less readable, but it has been proven to be also much more efficient.

```
  i = MATRIX_SIZE;
do {
  j = MATRIX_SIZE;
  do {
    k = MATRIX_SIZE;
    p1 = (int16_t *) p_m1;
    p2 = (int16_t *) p_m2;
    do {
        *p_res += (*p1) * (*p2);
        --k;
        ++p1;
        p2 += MATRIX_SIZE;
```

```
   } while (k > 0);
   --j;
   ++p_res;
   ++p_m2;
 } while (j > 0);
  --i;
  ++p_res;
  p_m2 -= MATRIX_SIZE;
  p_m1 += MATRIX_SIZE;
} while (i > 0)
```

In this piece of code, *p_res* always points to the position of the result matrix that is being computed, *p1* and *p2* are mobile pointers that move along the rows and the columns of the factor matrices in the inner loop, *p_m1* and *p_m2* are updated in the outer loops to point the currently active row and column (they are starting points for p1 and p2 at each iteration). Despite the extensive use of explicit pointer arithmetic, this way no additional multiplications are performed.

5.1.4 Results

The execution time of the matrix multiplication with both the implementations of the algorithm are reported here for different values of N.

N = 8:

- Theoretical number of operations: 512 multiplications + 512 sums;
- Simple implementation: 9255 μs;
- Optimised implementation: 3510 μs.

N = 16:

- Theoretical number of operations: 4096 multiplications + 4096 sums;
- Simple implementation: 73550 μs;
- Optimised implementation: 27528 μs.

N = 32:

- Theoretical number of operations: 32768 multiplications + 32768 sums;
- Simple implementation: 586983 μs;
- Optimised implementation: 198358 μs.

5.2 Performance improvements with Cortex M0+ architecture

5.2.1 Overview

Recently, Atmel released a new single chip microcontroller with integrated transceiver based on a 32 bit ARM Cortex-M0+ architecture (ATSAMR21E18A). The target of this SoC is similar to the ATMEGA256RFR2 used in the CIMC, so it might be the choice for a future transition from the CIMC to a final product[2]. This appendix contains some considerations about the improvements that the new microcontroller should bring to the

[2] As for now, there are no plans about it.

constraints described in Section 3.3 (keep in mind that these assumptions are very rough, as no actual testing was performed on the new SoC).

In summary, the information in the following should provide guidelines about which constraints are likely to be relaxed when the new M0+ architecture would be adopted.

5.2.2 Performance improvements

The Cortex-M0+ core is a 32-bit architecture, while the Atmega is an 8-bit one. Moreover, the clock of the ARM processor is faster, with a maximum of 48 MHz instead of 16 MHz. For these reasons, the new microcontroller is expected to perform much better in every benchmark, especially the more computation intensive ones – also if 48 MHz operation could have some impact on the average energy consumption.

5.2.2.1 Floating point multiplication and division

The ARM CPU can perform 32-bit multiplications in a single clock cycle. This capability should greatly improve the execution time of both the fixed and floating point multiplications, in particular the fixed multiplication routine tested in Section 3.3.1.5 should be executed in only 2 clock cycles (one for the multiplication and one for the shift). As for the division, it needs to be is performed at the software level and may take more or less cycles depending on the operands, but the ‘gcc’ documentation sets 95 cycles as an upper limit for 32-bit values.

5.2.2.2 Matrix multiplication

The matrix multiplication algorithm is computation intensive and involves many multiplications, so it benefits a lot from the advanced Cortex-M0+ architecture. The optimised algorithm presented in Section 3.3.1.8 should take about 12 clock cycles instead of 27:

• Fetching the values:	2×2 = 4 cycles;
• Performing operations (addition/multiplic.):	2 cycles;
• Storing the result:	2 cycles;
• Updating inner loop variables:	3 cycles;
• Overhead from outer loops:	≤ 1 cycle (less for larger matrices).

Moreover, switching to a matrix of 32-bit integers instead of 16-bit ones for more precision would cause a dramatic performance drop on the AVR, but no drop at all on the ARM processor.

5.2.2.3 Data block transfer

Since the Cortex-M0+ is a 32-bit architecture, it can load and store four bytes in a single 2 cycle instruction. This allows for a theoretical limit (supposing full loop unrolling) of 1 cycle/byte[3]. Even a simple loop copy algorithm would probably take no more than 1.5 cycles/byte, this means that the expected time required to copy a 128 byte buffer at 16 MIPS would be 12 µs, allowing full clock speed (48 MHz) the time drops to 4 µs.

[3] Assuming aligned memory blocks.

5.2.3 Power consumption improvements

According to the official documentation, the ATSAMR21E18A at 48 MHz has an average current consumption of 4.53 mA when running a standard algorithm (Fibonacci series computation was used in the test) that goes up to 6.32 mA for very intensive benchmark algorithms (CoreMark). These results are in the same range of the value expected from the current AVR microcontroller (Section 3.3.6), but since the ARM CPU is much faster, it would probably spend much more time in sleep state, leading to a significant improvement. Another power consumption improvement may be achieved by exploiting the more advanced sleep state of the ARM processor, for example the ‘Sleep Walk’ state that allows all the peripherals to run asynchronously without waking the CPU up. According to Atmel, the ATSAMR21E18A is very power efficient, consuming 50% less current than the current offerings.