Industrial Wireless Control Using Ultra-Wideband Radio

by

Daniel M. King

TR17-239, April 2017

This is an unaltered version of the author's MCS thesis
Supervisor: Bradford G. Nickerson

Faculty of Computer Science
University of New Brunswick
Fredericton, N.B. E3B 5A3
Canada

Phone: (506) 453-4566
Fax: (506) 453-3566
E-mail: fcs@unb.ca
http://www.cs.unb.ca

Abstract

This thesis investigates the use of ultra-wideband (UWB) radio for communication in real-time industrial wireless networks. OpenWSN is an IPv6 protocol stack based on IEEE 802.15.4e time slotted channel hopping (TSCH) media access control (MAC). The 802.15.4e standard describes MAC behaviors for common industrial communication settings such as process automation. We adapted OpenWSN to operate on an UWB physical layer using the DecaWave EVB1000 evaluation board hosting a DW1000 UWB transceiver.

WirelessHART is an existing technology for wireless process automation based on narrowband 2.4 GHz radio. We experimentally compared the performance of UWB and WirelessHART in two locations: an indoor office/laboratory environment and an industrial steam heating plant. In the office environment the UWB mesh network performed better than WirelessHART with an average of 95.53% packets successfully acknowledged over all guarateed time slots (GTSs), compared to an average of 71.21% for WirelessHART. In the industrial environment the performance of UWB was reduced to an average of 54.4% packets acknowledged for GTSs, compared to 84.55% for WirelessHART. The average observed upstream latency of all motes in the industrial environment was 149 ms for OpenWSN with UWB and 486 ms for WirelessHART.

Acknowledgements

I would like to thank my supervisors Dr. Bradford G. Nickerson and Dr. Wei Song for their guidance and suggestions in completing this thesis. Their ideas and suggestions greatly improved the work in this thesis and they have given me a memorable graduate experience.

Thanks also to Tom Gilmore, Cameron Dawson, and the staff at the UNB central heating plant for providing access to the plant on several occasions for the completion of the experimental part of this thesis.

Table of Contents

List of Tables

List of Figures

List of Symbols, Nomenclature or Abbreviations

$\wedge_{SL}$	Residual error rate of the safety communication layer
6LoWPAN	IPv6 over Low-power Wireless Personal Area Networks
AMCA	Adaptive Multi-Channel Adaptation
API	Application Programming Interface
ASN	Absolute Slot Number
ASN_{rx}^{AB}	ASN in which the packet from mote A is received by mote B.
ASN_{tx}^{AB}	ASN in which the packet from mote A is transmitted to mote B.
ASN_{tx}^{BA}	ASN in which the packet from mote B is received by mote A.
ASN_{rx}^{BA}	ASN in which the packet from mote B is transmitted to mote A.
B	Bandwidth
B_{frac}	Fractional bandwidth
BER	Bit Error Ratio
BERate	Bit Error Rate
BPM	Burst Position Modulation
BPSK	Binary Phase Shift Keying
C	Channel impulse response power
DAG	Directed Acyclic Graph
DSME	Deterministic and Synchronous Multi-Channel Extension
EIRP	Effective Isotropically Radiated Power
ε	Number of incorrect bits
f_c	Center frequency
f_H	Frequency upper bound
f_L	Frequency lower bound
FCC	Federal Communications Commission
FCS	Faculty of Computer Science
γ	Bit error rate limit
GTS	Guaranteed Time Slot
HART	Highway Addressable Remote Transducer
HRP	High Rate Pulse Repetition Frequency

IBER	Information Bit Error Rate
IPv6	Internet Protocol version 6
IR-UWB	Impulse Radio Ultra-Wideband
ISED	Innovation, Science and Economic Development Canada
LE	Low Energy
LLDN	Low Latency Deterministic Network
LRP	Low Rate Pulse Repetition Frequency
μ_d	Average downstream latency
μ_{RTT}	Average round trip time
μ_u	Average upstream latency
m	Number of recipients of a safety message
MAC	Medium Access Layer
n	Number of transferred bits
N_p	Number of received preamble symbols
N_{tx}	Number of packets transmitted
N_{txAck}	Number of packets for which an acknowledgement has been received
NLOS	Non Line of Sight
OOK	On Off Keying
O-QPSK	Offset Quadrature Phase-Shift Keying
Pe	Bit error probability
PAR	Packet Acknowledgement Ratio
PER	Packet Error Ratio
PHR	Physical Header
PLC	Programmable Logic Controller
PPM	Pulse Position Modulation
PRF	Pulse Repetition Frequency
R_{CRC}	Residual error probability of the CRC polynomial
RFID	Radio Frequency Identification
RPC	Remote Procedure Call
RPL	Routing Protocol for Low-Power and Lossy Networks
RS	Reed-Solomon
RTT	Round Trip Time
σ_d	Standard deviation of the downstream latency.
σ_{RTT}	Standard deviation of the round trip time.
σ_u	Standard deviation of the upstream latency.
SCL	Statistical Confidence Level
SDK	Software Development Kit
SECDED	Single Error Correction Double Error Detection
SFD	Start of Frame Delimiter
SIL	Safety Integrity Level
SoC	System on Chip
T_1	Time at which the request packet is generated.
T_2	Time at which the request is transmitted.
T_3	Time at which the request is received and the response is generated.
T_4	Time at which the response is received.

TCP	Transport Control Protocol
TH	Time Hopping
TSCH	Time Slotted Channel Hopping
UDP	User Datagram Protocol
UNB	University of New Brunswick
UWB	Ultra-Wideband
v	Maximum number of safety messages per hour
WPAN	Wireless Personal Area Network
WSAN	Wireless Sensor and Actuator Network
WSN	Wireless Sensor Network
XML	eXtensible Markup Language

Chapter 1

Introduction

Many industrial processes use control networks for automating a plant, process, factory, or machine. Wired networks are commonly used although wireless sensor networks (WSN) and wireless sensor and actuator networks (WSAN) are becoming increasingly popular as a way to reduce installation costs and increase flexibility [33]. WirelessHART [29] is one example of a successful wireless technology currently in use for industrial process automation.

Wireless communication faces several challenges that affect its reliability and performance. Industrial processes are often safety-critical where a failure in the system may cause significant harm or loss of life. IEC 61508 is a general standard for functional safety which defines requirements for the lifecycle of a safety-relevant system. Requirements and profiles for safe communication are defined in IEC 61784-3 [27] and IEC 61784-3-3 [28].

Ultra-wideband (UWB) radio is a different radio technology based on transmitting short pulses of RF energy which occupy a very wide bandwidth. The characteristics of UWB such as resistance to multipath fading and narrowband interference make it an interesting candidate for use in industrial wireless networks [24, 52]. In this thesis we investigate UWB in industrial wireless mesh networks and provide a comparison

against current technology used in industrial automation.

1.1 Research Objectives

The objectives of this thesis are to experimentally measure the performance of UWB in an industrial environment, and to compare the performance of an UWB-based mesh network with existing industrial WSN technology. Specific research questions addressed by this thesis are:

1. What is the reliability of UWB for real-time industrial communications?
2. How can IEEE 802.15.4 UWB be used in a real-time industrial WSN?
3. What is the relationship between UWB bit error rate (BERate) and the IEC 61508 safety integrity levels (SILs)?

1.2 Thesis Outline

Chapter 2 introduces different wireless technologies for industrial automation and provides an overview of the safety implications of communication. In Chapter 3 the performance metrics used to evaluate wireless networks are covered, and our adaptation of the OpenWSN protocol stack to an UWB physical layer is described. The experimental setup is described in Chapter 4, and the experimental results are presented in Chapter 5. Finally, Chapter 6 contains the conclusions and ideas for future work.

Chapter 2

Background

2.1 Wireless Industrial Control Networks

Many industrial processes use control networks for automating a plant, process, or machine. The control networks consist of various sensors and actuators, connected to control logic via a network. Control networks can vary in size, from a few sensors and actuators in a single machine, to many hundreds in a large plant. Wireless networks are an attractive alternative to wired networks for industrial process automation as they can offer reduced installation cost and greater flexibility [33].

Different network topologies are possible in industrial WSNs to meet different requirements. The different network topologies are shown in Figure 2.1 In a star topology each device (called a "mote") communicates directly with the gateway which provides the lowest possible latency since packets are not routed through intermediate motes. A hybrid mesh network extends the operational range of the network by using additional motes that are responsible for routing packets between the sensors and the gateway. The sensors do not route messages and can remain low-power. In a mesh network every device is capable of routing messages which provides increased redundancy to improve communication reliability.

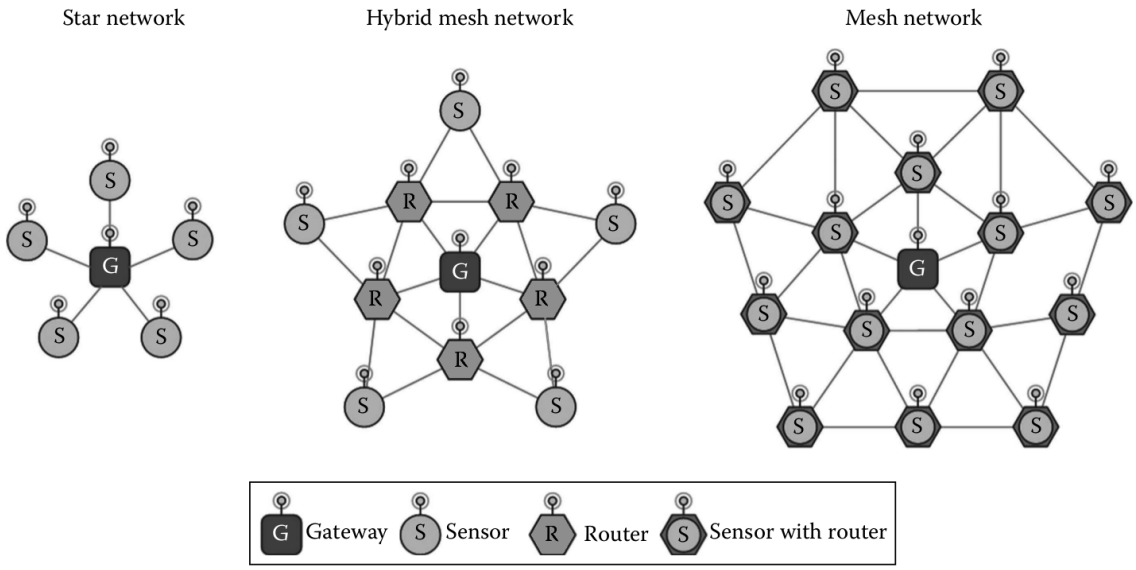


Figure 2.1: WSN topologies (from [33]).

Wireless networks are exposed to various conditions which reduce their reliability compared to wired networks. Some of the challenges faced by WSNs are as follows [23, 33, 10]:

- *Harsh environment.* Industrial environments may expose devices to extreme temperatures, humidity, and vibration. Wireless networks may be exposed to multipath fading, slow/fast fading, and varying signal power.

- *Electromagnetic interference (EMI).* Electrical equipment such as drives and welding may cause interference in RF bands.

- *Packet errors.* Wireless communication generally experience higher BERate (10^{-2} to 10^{-6}), and may vary over time due to changes in the environment such as moving obstacles.

- *Interference from other wireless networks.* Other wireless networks such as ZigBee, WiFi, and Bluetooth can cause interference with wireless networks that use the same frequency band.

- *Power consumption.* Devices in wireless networks are often battery powered

and it is desirable for the devices to use at little power as possible in order to prolong the battery life. Some devices must meet the requirement for a more than 10 year battery life.

2.1.1 WirelessHART

IEC standard 62591 [29] specifies the WirelessHART protocol for wireless industrial process control. WirelessHART is designed as a wireless extension to the existing highway addressable remote transducer (HART) protocol used in wired fieldbuses. Consequently, the application layer is similar for wired and wireless HART.

The WirelessHART physical layer uses the 2.4 GHz offset quadrature phase-shift keying (O-QPSK) physical layer defined in IEEE 802.15.4-2006. The media access control (MAC) layer is based on IEEE 802.15.4-2006 data frames, and utilizes time division multiple access (TDMA) and frequency hopping. The full WirelessHART protocol stack is shown in Figure 2.2.

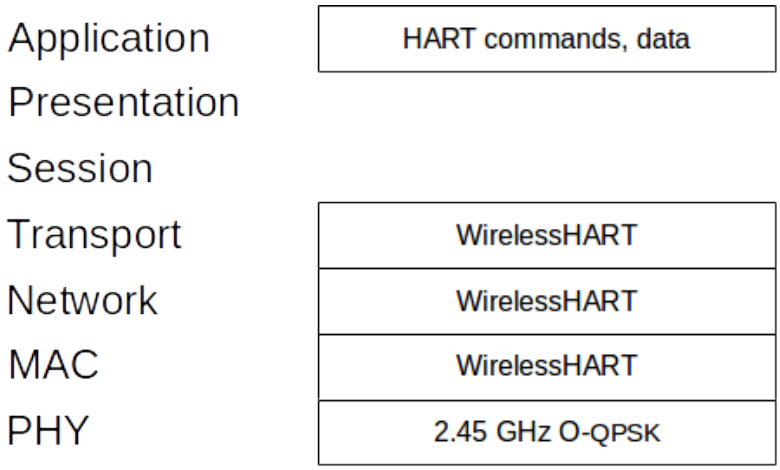


Figure 2.2: The WirelessHART protocol stack (adapted from [20]).

.

The network layer is designed for a power-optimized redundant mesh network to ensure reliable transfer of data [20]. The transport layer provides reliable stream transport of end-to-end acknowledged packets, as well as best effort communication

without end-to-end acknowledgements.

The WirelessHART network architecture is presented in Figure 2.3. The motes communicate wirelessly to form the mesh network to the network manager via an access point. The network manager is responsible for configuration of the network and the management of routing messages and monitoring of the network conditions.

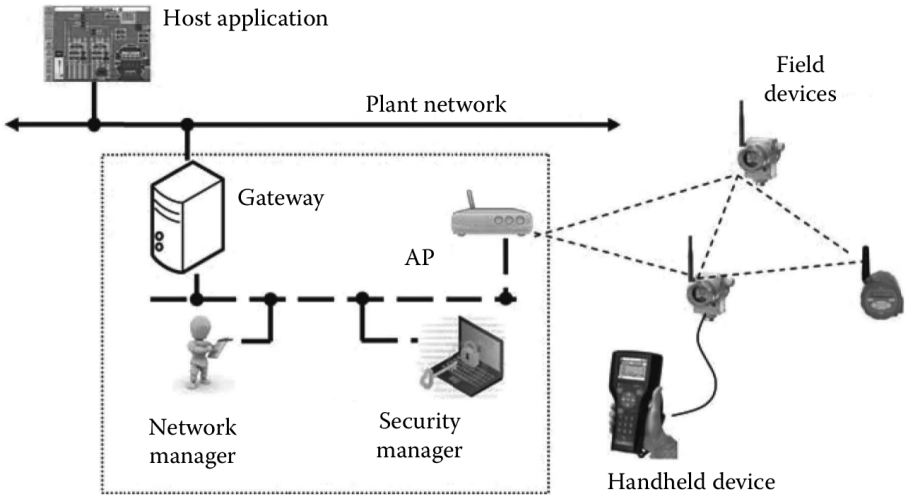


Figure 2.3: The WirelessHART network architecture (from [20]).

Channel hopping is used in the MAC layer to provide robustness in the presence of poor conditions on certain channels, as a failed transmission can be retried on another channel. Furthermore, channels may be manually blacklisted to prevent their use by the WirelessHART MAC layer, which can avoid interference on certain channels from other users of the frequency band, such as WiFi.

2.1.2 ISA100.11a

ISA100.11a is another industrial wireless network based on the same IEEE 802.15.4 2.4 GHz physical layer as WirelessHART. A key difference from WirelessHART is that the ISA100.11a protocol stack is based on the internet protocol version 6 (IPv6). The ISA100.11a protocol stack is shown in Figure 2.4. A comparison between WirelessHART and ISA100.11a can be found in [31, 39].

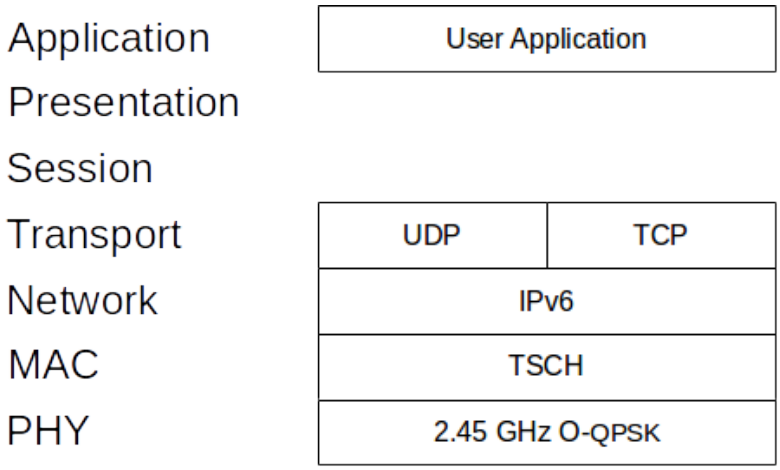


Figure 2.4: The ISA100.11a protocol stack (adapted from [41]).

.

2.2 Safety Aspects of Industrial Control Networks

Some industrial control systems are responsible for controlling a plant or process where failure may result in the system causing harm to humans or the environment [45]. Some examples of such applications include hydraulic presses, robots, chemical processes, and fire and gas sensing.

Industrial control networks in a safety-critical system are responsible for transporting messages which contain data relevant to the safe operation of the system. A failure in the communucation of a safety-related message such as corruption of the message may cause the system to enter an unsafe state where a dangeous failure occurs.

2.2.1 IEC 61508

IEC 61508 [26] is a general standard for functional safety which defines rules throughout the system lifecycle based on a risk management approach. The definition of safety according to the standard is the "freedom from unacceptable risk". During the design of a safety system conforming to IEC 61508 the risks must be identified and assessed based on the probability and impact of their occurrence. IEC 61508 risk categories and the relationship of IEC 61508 to other industry-specific standards

are discussed in [31].

A dangerous failure occurs when the system enters an unsafe state which may cause harm to humans or the environment. The definition of a dangerous failure is dependent on the specific system and must be carefully defined. IEC 61508 defines four *safety integrity levels* (SIL) which associate an acceptable level of risk for a safety system, with SIL 4 as the most dependable and SIL 1 as the least dependable. Table 2.1 shows the dangerous failure rate limits imposed by the four SILs.

Table 2.1: IEC 61508 safety integrity levels (from [26]).

SIL	High demand rate Probability of dangerous failure per hour of operation (PFH)	Low demand rate Probability of dangerous failure on demand (PFD)
1	$\geq 10^{-6}$ to $\leq 10^{-5}$	$\geq 10^{-2}$ to $\leq 10^{-1}$
2	$\geq 10^{-7}$ to $\leq 10^{-6}$	$\geq 10^{-3}$ to $\leq 10^{-2}$
3	$\geq 10^{-8}$ to $\leq 10^{-7}$	$\geq 10^{-4}$ to $\leq 10^{-3}$
4	$\geq 10^{-9}$ to $\leq 10^{-8}$	$\geq 10^{-5}$ to $\leq 10^{-4}$

Each safety function belongs to one of two modes defined by IEC 61508 based on the frequency of which the safety function is used. Low demand functions are invoked less than once per annum, and high demand functions are invoked at least once per annum. An avionics fly-by-wire control system is an example of a high demand mode system, and a car airbag is an example of a low demand mode system. As shown in Table 2.1, high demand mode systems are required to have a lower probability of a dangerous failure per hour of operation.

2.2.2 IEC 61784-3

In any communication network there are a variety of possible errors that can occur which negatively affect the messages sent across the network, such as packet loss or corruption. In a safety-related system the risk of communication errors must be included when considering the overall risk of a system. IEC 61784-3 [27] introduces

safety aspects for the communication of messages containing safety-related information.

There are two classes of errors that can occur within a communication network: *deterministic* and *stochastic* errors. The deterministic errors identified by IEC 61784-3 are [27]:

- *Corruption*: A message is corrupted due to transmission errors or interference.
- *Unintended repetition*: A message is repeated at an incorrect point in time.
- *Incorrect sequence*: Messages arrive at their destination in the incorrect order.
- *Loss*: A message is never received or acknowledged.
- *Unacceptable delay*: A message is delayed beyond an acceptable time window.
- *Insertion*: A message is inserted from an unexpected or unknown source.
- *Masquerade*: A message is inserted from an incorrect but apparently valid source. This may cause a non-safety related message to be received and processed by a safety-related entity.
- *Addressing*: A safety-related message is sent to the incorrect safety-related entity.

To address these errors, IEC 61784-3 defines several remedial measures [27]:

- *Sequence number*: A sequence number is added to each message that is sent between a source and destination. The sequence number changes for each message in a predetermined way (typically by incrementing the sequence number for each message).
- *Time stamp*: Many messages are only valid for a certain period of time. A time stamp is added to each message to allow the recipient to determine whether or not the message is still valid.

- *Time expectation*: The message destination expects a message to be received within a certain period of time. If the message is not received within this time then an error is assumed. Watchdog mechanisms fall into this category.

- *Connection authentication*: Unique source and destination identifiers are added to each message to identify the relevant safety-related entities.

- *Feedback message*: Also known as aknowledgements. The receiver of a message sends a feedback message to the source to acknowledge correct receipt of a message.

- *Data integrity assurance*: Redundant data is added to each message to permit detection of data corruption.

- *Redundancy with cross-checking*: A message is sent more than once, possibly using different media and protocols. The receiver cross-checks the messages and if there is a difference then an error is assumed.

Each remedial measure addresses one or more types of deterministic error. Table 2.2 maps shows the deterministic errors that are addressed by each remedial measure.

2.2.2.1 Relationship with IEC 61508 SIL

Stochastic errors in a communication system, such as inteference, can cause one or more bits in a message to be pertubed during transmission. Communication protocols usually use a data integrity assurance mechanism to detect the presence of incorrect bits in a message. A common technique is cyclic redundancy check (CRC). CRC algorithms are good at providing protection against stochastic errors, but they cannot detect all possible combinations of bit errors that can occur in a message, including the CRC bits. Certain combinations of bit errors in a message can result in a corrupted message being received which has an apparently valid CRC. The

Table 2.2: Deterministic communication errors and their remedial measures (adapted from [27]).

	Sequence number	Time stamp	Time expectation	Connection authentication	Feedback message	Data integrity	Redundancy with cross-checking
Corruption					X	X	
Unintended repetition	X	X					X
Incorrect sequence	X	X					X
Loss	X				X		X
Unacceptable delay		X	X				
Insertion	X			X	X		X
Masquerade				X	X		
Addressing				X			

situation where a corrupted message is received, but is not detected as incorrect by the error checking mechanisms, is known as a *residual error*.

Generally, the residual error probability increases as the bit error probability increases. However, the characteristics of the CRC polynomial also affects the residual error probability. For a *proper* polynomial of an r-bit CRC the residual error probability approaches, but does not exceed the limit 2^{-r} as the bit error probability increases. For an *improper* polynomial the residual error probability exceeds 2^{-r} for some bit error probabilities. The relationship between the residual error probability and bit error probability for proper and improper polynomials is shown in Figure 2.5. IEC 61784-3 defines the relationship between the bit error probability and residual error rate, as shown in Equation 2.1 [27].

$$\wedge_{SL}\ (Pe) = R_{CRC}\ (Pe) \times v \times m \tag{2.1}$$

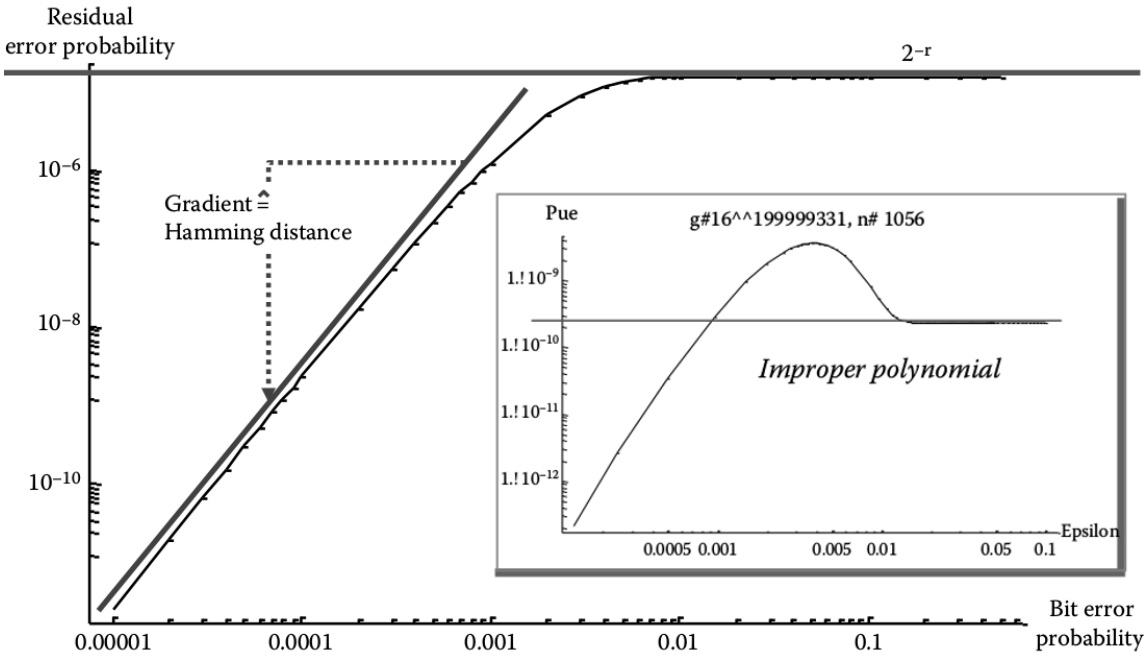


Figure 2.5: Relationship between bit error probability and residual error probability for proper and improper polynomials with $r = 16$ (from [45]). Pue means the probability of undetected error, and epsilon is the bit error probability.

where [27]:

- Pe is the bit error probability. A value of 10^{-2} is used unless a better bit error probability can be proven.
- $\wedge_{SL}$ (Pe) is the residual error rate per hour of operation of the safety communication layer with respect to the bit error probability Pe.
- R_{CRC} (Pe) is the residual error probability of the CRC with respect to the bit error probability Pe.
- v is the maxmimum number of safety messages per hour.
- m is the maximum number of recipients of the message between the source and destination of the safety function.

IEC 61784-3 requires that the residual error rate does not exceed 1% of the target SIL, which is known as the *1% rule*. For example, to achieve SIL 3 the residual error

rate $\wedge_{SL}$ cannot exceed 10^{-9}.

2.2.3 PROFIsafe

Although many existing industrial networking technologies are suitable for real-time industrial control, they are not suitable to be directly depended upon in safety critical applications where there is significant risk since they do not meet the strict requirements defined in IEC 61508 [45].

PROFIsafe is an application layer protocol defined in IEC 61784-3-3 [28] which implements the mechanisms defined by IEC 61784-3 in order to achieve certification up to SIL 3. The PROFIsafe safety layer does not rely on the mechanisms of the lower protocol layers, instead relying on its own error checking mechanisms. This approach is called the *black channel* and is illustrated in Figure 2.6.

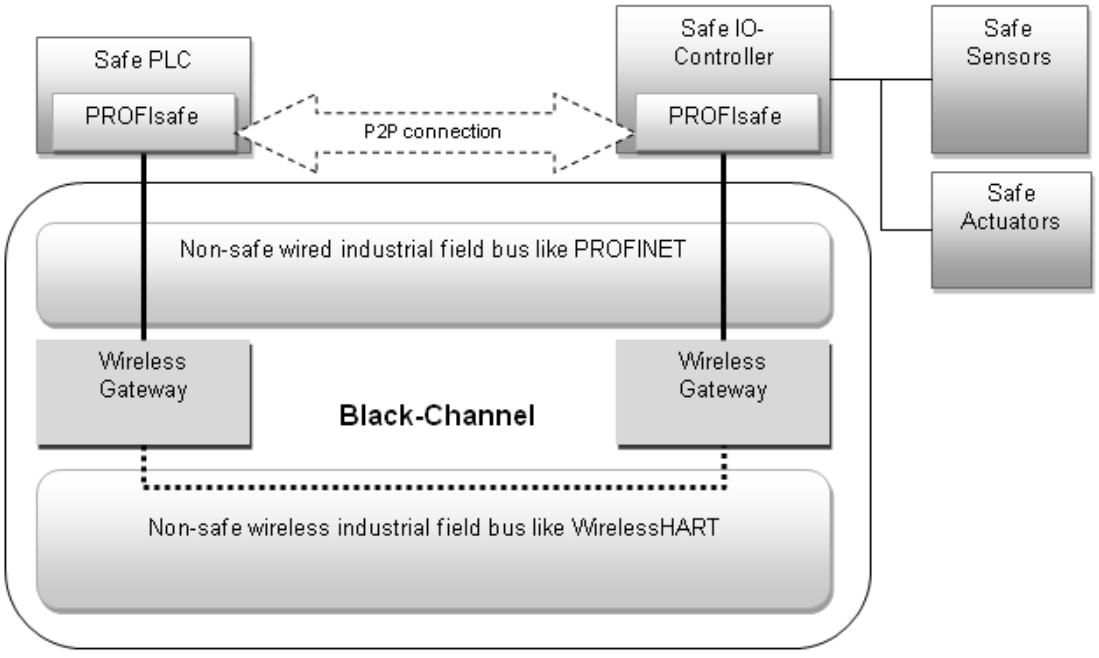


Figure 2.6: The black channel model (from [8]).

With the black channel approach, the safety layer does not assume any guarantees provided by the lower layers, such as the residual error probability. Instead, the

safety layer implements all required mechanisms to meet the target SIL alone. This approach permits safety-relevant systems to use existing industrial networks and coexist alongside non safety-relevant systems on the same network, i.e. "what he/she is using, he/she will not be losing" [45].

Section 2.2.2.1 defined the relationship between the BER of a communication network to the SILs. Although PROFIsafe implements its own error checking mechanisms such as a CRC, the bit error probability of the underlying media affects the ability of PROFIsafe to meet the residual error rate requirements for the target SIL. If the bit error probability is too high then the probability of a residual error in the PROFIsafe layer will exceed the limit permitted by the target SIL. For SIL 3 the bit error probability must not exceed 10^{-2} [8].

2.2.3.1 PROFIsafe in Wireless Networks

The black channel model permits PROFisafe to be used on different communication media including wireless networks, provided that the BERate is low enough for the target SIL. PROFIsafe over ISA100.11a is shown in [40] in a battery-powered gas detector certified to the SIL 2 low demand mode.

PROFIsafe over WirelessHART is investigated experimentally by Åkerberg *et. al.* in [7, 8]. The authors measure the network latency and packet error rate of a PROFIsafe device operating in a WirelessHART network in an industrial heating and power production plant. They found that although safe communication with PROFIsafe was possible in principle, the harsh RF environment caused long latency in the network, resulting in fail-safe timeouts in the PROFIsafe application.

2.3 Ultra-Wideband Radio

Ultra-wideband (UWB) radio is defined by the Federal Communications Commission (FCC) as an intentional radiator that has either a fractional bandwidth (B_{frac}) greater than 0.20 or a bandwidth greater than or equal to 500 MHz [13]. The frequency upper and lower bounds (f_H and f_L respectively) are the points that are 10 dB below the highest radiated power part of the band. The bandwidth B is defined as:

$$B = f_H - f_L \tag{2.2}$$

The center frequency f_c is the center part of the band, defined as:

$$f_c = f_L + \frac{f_H - f_L}{2} \tag{2.3}$$

The fractional bandwidth, B_{frac} is defined as:

$$B_{frac} = \frac{2(f_H - f_L)}{f_H + f_L} \tag{2.4}$$

The power spectral density of an UWB signal is shown in Figure 2.7.

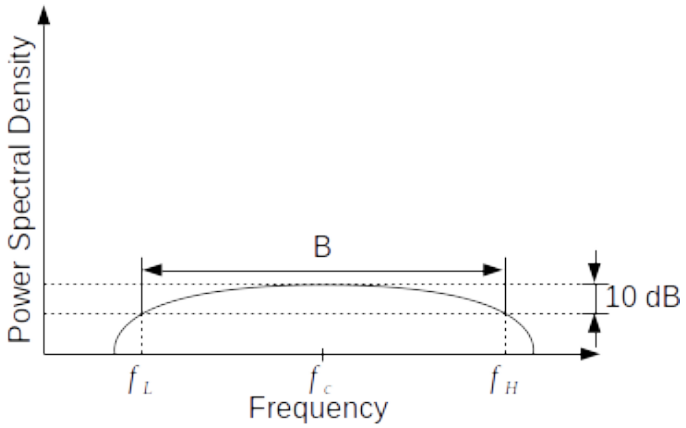


Figure 2.7: Power spectral density of UWB radio (adapted from [25]).

2.3.1 Impulse Radio Ultra-Wideband

Impulse radio ultra-wideband (IR-UWB) is one method for an UWB system which is based on the transmission of short pulses of RF energy. The relationship between the pulse duration (T_c) and the bandwidth (B) is defined as [25]:

$$B = \frac{1}{T_c} \tag{2.5}$$

So, in an UWB system with a bandwidth of at least 500 MHz the pulse duration is no greater than 2 ns. The rate at which pulses are transmitted in an IR-UWB system is known as the pulse repetition frequency (PRF).

2.3.2 Modulation Techniques

This section describes IR-UWB modulation schemes that are used by the UWB physical layer described in Section 2.4.1. Different modulation schemes have different BERate characteristics in the presence of noise, interference, and multipath fading. The choice of modulation techinque therefore has an impact on the achievable SIL through the relationship defined in Equation 2.1.

2.3.2.1 Pulse Position Modulation

Pulse position modulation (PPM) is a technique which encodes binary data based on the timing of transmitted pulses. In a basic PPM scheme, each symbol is split into two halves, each of which has a duration of half of the symbol duration (T_{sym}). The binary value is encoded by which half contains the transmitted pulse. For example, a binary 0 may be encoded by transmitting the pulse in the first half of the symbol period, and a binary 1 as transmitting in the second half of the symbol period.

A PPM scheme can also support multiple users by further dividing each half of the symbol period into an equal number of slots and utilizing a time hopping (TH)

code to vary which slot contains the transmitted pulse for each symbol. Users don't interfere with each other providing that they use different slots for each symbol that they transmit.

The BERate performance of PPM over multipath channels is investigated by Ge *et. al.* in [21]. Their results show that the performance of PPM deteriorates when there exist multipath components with a delay matching the PPM time shift. The results also show that the probability of a bit error is asymmetric for a 1 versus a 0.

The number of possible pulse positions within a symbol also affects the BERate, as shown by Abedi and Yagoub in [5]. Their model assumed additive white Gaussian noise and showed that the BERate is reduced as the number of possible pulse positions increases.

Burst position modulation (BPM) is a variant of PPM where a burst of several pulses are transmitted instead of only one pulse.

2.3.2.2 Binary Phase Shift Keying

Binary phase shift keying (BPSK) modulation encodes binary data based on the polarity of the transmitted pulse (positive or negative). For example a binary 1 is encoded with a pulse with positive polarity, and a binary 0 is encoded as a pulse with negative polarity.

2.3.3 Regulation and Power Limits

Due to the wide bandwidth of UWB systems there is the risk that the UWB system may interfere with narrowband systems over which the UWB systems' bandwidths overlap. To prevent such interference, UWB is regulated by organizations such as the FCC [13] in the United States, and Innovation, Science and Economic Development Canada (ISED) [1] in Canada. These organizations define limits for the equivalent isotropically radiated power (EIRP) of UWB, which does not exceed

−41.3 dBm/MHz. Figure 2.8 shows the EIRP limit for frequencies up to 12 GHz, as defined by ISED and the FCC.

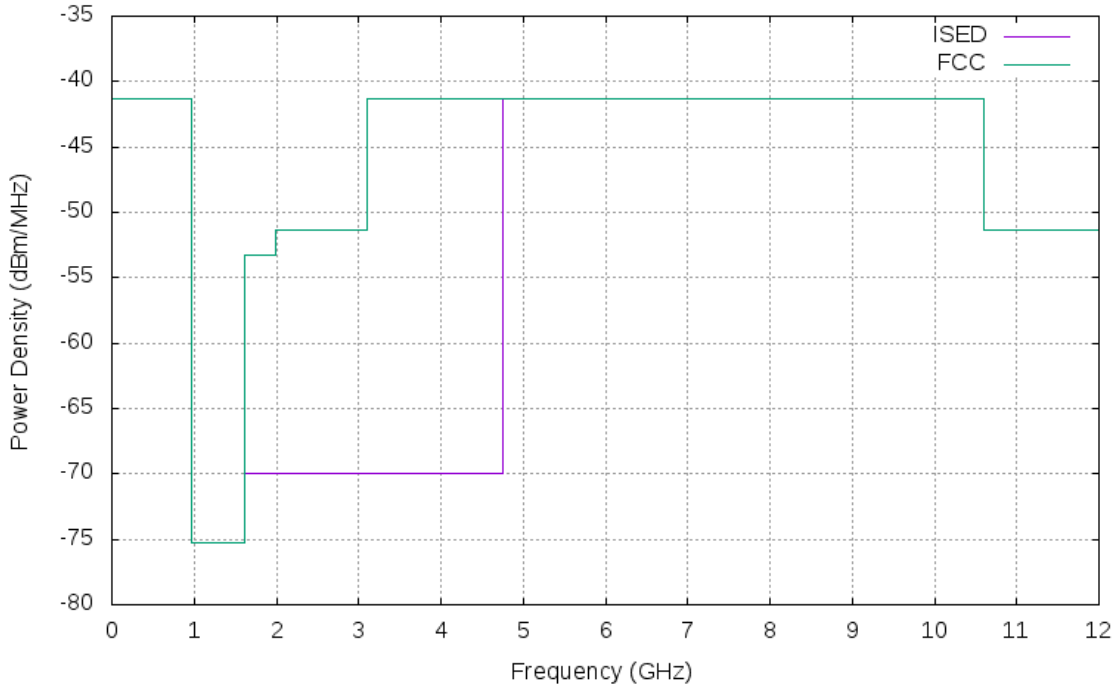


Figure 2.8: ISED and FCC UWB EIRP limits for indoor communication systems (adapted from [1, 13]).

2.3.4 Advantages of UWB

Some of the advantages of UWB communication are [22, 25]:

- The extremely short duration of UWB pulses ($\leq$ 2 ns) results in strong resistance to multipath fading. The use of rake reception in the UWB receiver can reduce the BERate in multipath environments [34].
- Resistance to interference from narrowband radios due to the high channel bandwidth [53].
- The low power spectral density permits UWB to share spectrum without causing interference.

- UWB signals are resistant to interception and eavesdropping due to the low average transmission power.
- High data rates up to 480 Mbps can be achieved with UWB systems.

2.4 IEEE 802.15.4

IEEE 802.15.4 is a standard for low-rate wireless personal area networks (LR-WPAN). The standard defines several physical layers operating in various frequency bands, and defines the MAC layer to operate the WPAN.

Since its introduction in 2003, IEEE 802.15.4 has been augmented and updated to include new physical layers, and to expand upon the functionality of the MAC layer. The current latest approved version is IEEE 802.15.4-2015 [3], and the latest amendment is IEEE 802.15.4q-2016 [4]. A list of notable amendments related to this thesis are shown in Table 2.3.

Table 2.3: Notable versions and amendments of the IEEE 802.15.4 standard.

Version	Description
IEEE 802.15.4-2003	First version.
IEEE 802.15.4a-2007	UWB physical layer is introduced.
IEEE 802.15.4e-2012	New MAC sublayer behaviours are introduced.
IEEE 802.15.4f-2012	Low rate PRF (LRP) UWB physical layer introduced.
IEEE 802.15.4-2016	Latest version.

2.4.1 UWB Physical Layer

Two UWB physical layers are defined by IEEE 802.15.4: high-rate PRF (HRP) UWB, and low-rate PRF (LRP) UWB. The HRP UWB was first introduced in the IEEE 802.15.4a-2007 amendment, and the LRP UWB physical layer was introduced in the IEEE 802.15.4f-2012 amendment. This thesis is based on the HRP UWB physical layer.

Although the HRP and LRP UWB physical layers are both based on IR-UWB, they are quite different in their design and capabilities. For example, the HRP UWB physical layer is capable of precision ranging between devices, whereas the LRP UWB physical layer is not. A comparison between the HRP and LRP UWB physical layers is shown in Table 2.4.

Table 2.4: Comparison of HRP UWB and LRP UWB features.

Feature	LRP UWB	HRP UWB
Data Rates	31.25 kbps 250 kbps 1 Mbps	110 kbps 850 kbps 6.81 Mbps 27.24 Mbps
Peak PRF	2 MHz	499.2 MHz
Ranging	No	Yes
Multi-user interference supression	No	Yes
Modulation	OOK PPM	BPSK BPM
Error Correction	SECDED Convolutional	SECDED Convolutional Reed-Solomon

2.4.1.1 HRP UWB Frame Structure

The physical frame is composed of three parts, shown in Figure 2.9: the SYNC field, the physical header (PHR) and the data field. The SYNC field is further split into two parts: the preamble and start of frame delimiter (SFD).

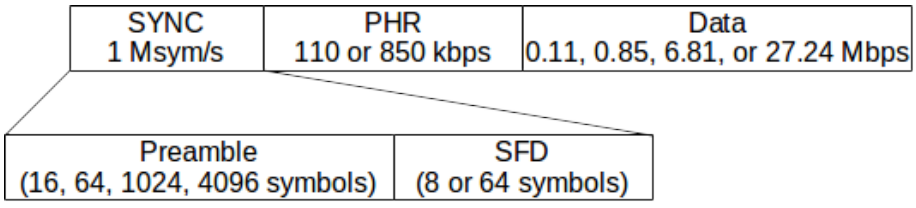


Figure 2.9: Structure of an UWB physical frame (adapted from [3]).

The SYNC field allows the receiver to detect the presence of the radio packet and synchronize with the transmitter. The PHR contains information about the data to be received, including the length of the data and the data rate used to transmit the data. The data field contains the encoded user data.

Unlike most other physical layers defined in IEEE 802.15.4 the three parts of the HRP UWB physical frame can be transmitted at different data rates. The SYNC field is transmitted at a base rate of either approximately 1 Msym/s for 16 MHz and 64 MHz PRFs, or 0.25 Msym/s for the 4 MHz PRF. The PHR is transmitted at either 110 kbps if the data rate is 110 kbps, or 850 kbps for data rates of 850 kbps, 6.81 Mbps, and 27.24 Mbps.

2.4.1.2 HRP UWB Error Correction

Three error correction schemes are used by the HRP UWB physical layer. A single error correction double error detection (SECDED) algorithm is used to protect the physical header portion of the physical frame. The data portion of the frame is protected with a combination of Reed-Solomon (RS) [49, 46] and a 1/2 rate systematic convolutional code [46].

The user data is first encoded using RS which adds an extra 48 parity bits to the data. The output of the RS encoding is then encoded using the convolutional code. The resulting encoded data forms the data part of the physical frame.

The convolutional code outputs two bits. The first bit determines the position of the burst used in BPM, and the second bit determines the burst polarity in the BPSK modulation. For the 27.24, Mbps data rate the convolutional code is not used.

The use of RS and convolutional coding alongside the wide bandwidth of UWB radio allow the communication reliability to come closer to the theoretical Shannon capacity of the channel in a noisy environment. A performance analysis of the coding schemes used in IEEE 802.15.4a UWB can be found in [6].

2.4.2 MAC Layer

The IEEE 802.15.4 MAC layer defines several configurations for the MAC layer. For example, the WPAN may be beacon enabled or non-beacon enabled which permits slotted or non-slotted access, and use guaranteed time slots (GTS) or only contention access among motes, among other configuration options.

IEEE 802.15.4e-2012 greatly expanded the capability of the MAC layer by introducing several new MAC modes:

- Time slotted channel hopping (TSCH);
- Low latency deterministic network (LLDN);
- Deterministic and synchronous multi-channel extension (DSME);
- Aynchronous multi-channel adaptation (AMCA);
- Low energy (LE); and
- Radio frequency identification (RFID).

The MAC modes of interest to industrial automation are: TSCH, LLDN, and DSME. LLDN is designed for low latency applications and permits the cycle time to be tightly controlled by configuring the time slot duration and the number of nodes in the network. LLDN networks operate in a star topology on a single channel. A multi-channel adaptation for LLDN is presented in [38].

DSME is designed for scalable and reliable networks which require deterministic latency. It is based on the mesh network topology. The target applications for DSME are: process automation, factory automation, and smart metering.

TSCH is designed for process automation and is also based on the mesh network topology. It is described in detail in the following section.

2.4.2.1 Time Slotted Channel Hopping

The TSCH MAC mode is based on the time synchronization technology developed by WirelessHART. Consequently, TSCH shares many similarities with the WirelessHART MAC layer. A comparison between IEEE 802.15.4e TSCH and WirelessHART can be found in [11].

TSCH utilizes time division multiple access (TDMA) and frequency division multiple access (FDMA) techniques. Channel access is divded into *time slots* of a fixed duration during which a data exchange occurs between each mote. During each time slot the communication can occur on one of several channels. This permits multiple devices to communicate in the same time slot without contention, provided that the devices use different channels.

The TSCH MAC mode defines a *slotframe* as a repeating sequence of an arbitrary number of time slots. A time slot is long enough to permit two devices to exchange a single data packet and an acknowledgement. The number of time slots in the slotframe determines how often two devices can communicate. Devices can be allocated multiple slots within a slotframe if they require more frequent communication than the duration of the slotframe would otherwise permit. The allocation of slot communication is handled by the upper layer. An example of a slotframe is shown in Figure 2.10.

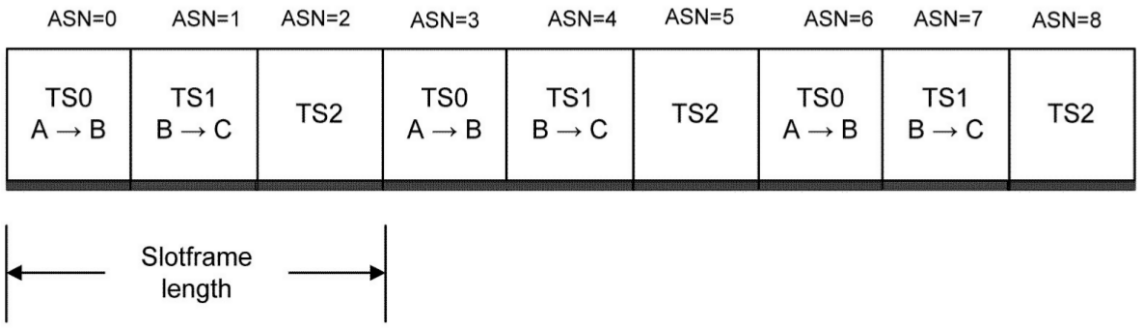


Figure 2.10: An example TSCH slotframe (from [30]).

A network may contain multiple slotframes with different lengths, which permits

some devices on the network to have more frequent communication using a short slotframe, while other devices have less frequent communication on a longer slotframe. Figure 2.11 shows an example of multiple slotframes.

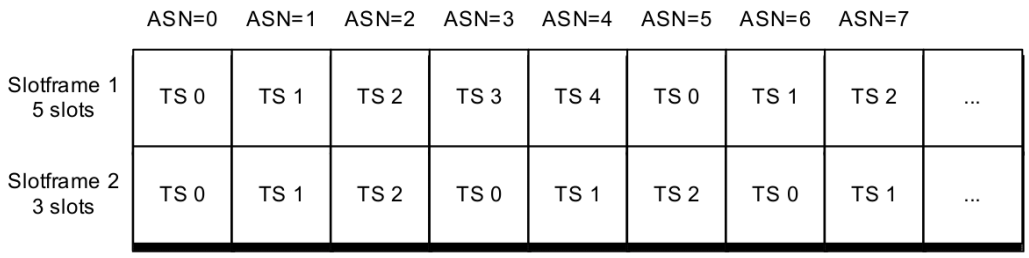


Figure 2.11: An example TSCH multiple slotframe (from [30]).

The physical layer channels used for communication in TSCH changes for each time slot independently of the slotframe length, known as channel hopping. Channel hopping permits each device to use all available physical channels for communication and ensures reliable data exchange even in the presence of interference on one or more channels. The choice of channel to use depends on the current absolute slot number (ASN) and the *channel offset* assigned to the device. Multiple devices can transmit in the same time slot provided that they have a different channel offset.

Devices in a TSCH network are synchronized to the start of each time slot so that each device begins a new time slot at the same time. This is achieved by time synchronization with neighboring devices, or the PAN coordinator if it is in range. Neighboring devices that are used for time synchronization are called *time synchronization parents*. Data packets sent in a time slot to a time synchronization parent are expected to be received at a fixed offset from the start of the time slot, denoted as *macTsTxOffset*. When the two devices are not perfectly synchronized then the message will be received before or after *macTsTxOffset*. The time synchronization parent measures the error and returns *time correction* information in the acknowledgement packet to allow the originator device to correct for the clock drift. This mechanism permits devices to remain synchronized to well within 1 ms. Further details of TSCH time synchronization can be found in [3].

Chapter 3

Ultra-Wideband in Wireless Control

3.1 Performance Metrics

3.1.1 Bit Error Rate

Section 2.2.2 defined the relationship between the bit error probability Pe and the IEC 61508 SIL in Equation 2.1. If the Pe is too high then the target SIL cannot be met. For example, at SIL 3 the Pe cannot exceed 10^{-2} [8].

The bit error ratio (BER) is defined in Equation 3.1 [43] as the total number of incorrect bits ε divided by the total number of transferred bits n in a specific time period. The BER approaches the true bit error probability Pe of the channel as the number of bits transferred through the channel approaches infinity.

$$BER = \frac{\varepsilon}{n} \xrightarrow[n\to\infty]{} Pe \tag{3.1}$$

For receivers utilizing error correction schemes the BER is also defined as [18]:

- The transmission bit error ratio (TBER) is the total number of incorrect bits

before error correction divided by the total number of transferred bits, including redundant error correcton codes.

- The information bit error ratio (IBER) is the total number of bits that are incorrect after error correction divided by the total number of transferred bits.

For specifications such as the IEC 61508 SILs there exists a Pe limit γ for the communication channel which must not be exceeded, e.g. 10^{-2} for SIL 3. The statistical confidence level (SCL) that the true bit error probability Pe is below the limit γ for a given BER measurement ε and n can be expressed mathematically as shown in Equation 3.2 [43, 36] which can be solved empirically using a computer.

$$SCL = P[Pe < \gamma | \varepsilon, n] = 1 - \sum_{k=0}^{\varepsilon} \frac{n!}{k!(n-k)!} \gamma^k (1-\gamma)^{n-k} \tag{3.2}$$

To accurately estimate the true Pe from the measured BER a large number of bits must be transferred through the channel. Since testing with more bits takes more time there is a trade-off between the SCL and the measurement time. The estimated number of bits that must be transferred to determine whether $Pe < \gamma$ to a given SCL, given ε incorrect bits, is derived in [43] and shown in Equation 3.3.

$$n = -\frac{ln(1-SCL)}{\gamma} + \frac{ln\left(\sum_{k=0}^{\varepsilon} \frac{(n\gamma)^k}{k!}\right)}{\gamma} \tag{3.3}$$

where $n\gamma$ must be greater than 1, i.e. at least as many bits are transferred as the inverse of the target limit γ, and k is the same order of magnitude as $n\gamma$ [36].

3.1.2 Packet Error Ratio

The bit error rate defines the rate at which bit errors are expected to occur in a sequence of bits transmitted through a channel. In reality, data is transmitted as a finite sequence of bits in packets. Even a single incorrect bit in a decoded packet

causes a *packet error* to occur.

The packet error ratio is the number of packet errors ε_p divided by the total number of packets received n_p, as shown in Equation 3.4.

$$PER = \frac{\varepsilon_p}{n_p} \tag{3.4}$$

Assuming a uniform distribution of incorrect bits, the probability of a packet error P_p for l-bit packets can be calculated from the bit error probability as shown in Equation 3.5 [44].

$$P_p = 1 - (1 - Pe)^l \tag{3.5}$$

Equation 3.5 does not model all environments or networks, as the bit errors may occur in bursts instead of following a uniform distribution. The presence of burst errors would reduce the PER as incorrect bits would be grouped together in fewer packets.

3.1.3 Packet Loss Ratio and Latency

Many industrial processes require messages to be received in a timely fashion in order to operate correctly. For example, if sensor readings arrive too late then they may not reflect the current state of the system. Similarly, messages that fail to arrive may reduce system performance. Packet loss and latency in a WSAN are influenced by a variety of factors including: low-power transmission, variable transmit power, multi-hop transmission, noise, radio interference, and node mobility [50]. Figure 3.1 shows how the packet loss ratio (PLR) can drastically change from 0% to 100% over distances as short as 10 m.

The number of packets lost is calculated by subtracting the number of received packets from the total number of transmitted packets. The PLR is calculated as the

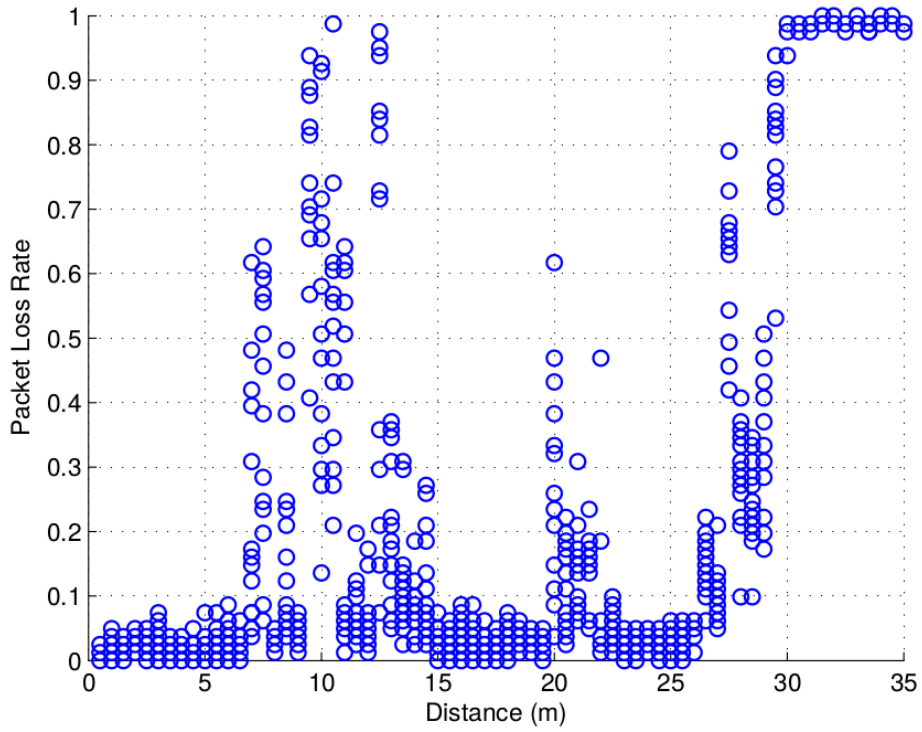


Figure 3.1: Packet loss rate versus distance from real-world measurements (from [50]).

number of lost packets divided by the number of transmitted packets [32].

The system error under different PLRs using a distributed control scheme is reported in [12]. Their results show that the system error with a PLR of 20% is close to the system error with a PLR of 0%, and the system is able to perform with only slightly higher system error with a PLR of 40%. At a PLR of 80% the control performance is severely degraded.

The safety function response time (SFRT) is defined in IEC 61784-3-3 [28] as the worst case time elapsed from the actuation of a safety actuator to when the safety actuator enters the safe state, even in the presence of errors. A SFRT model is defined for single input single output systems in IEC 61784-3-3. The SFRT model is adapted in [42] for multiple input multiple output systems. The SFRT must account for the maximum packet latency in the network, so longer latencies have a direct impact on the SFRT. For systems based on discrete time steps, packets that arrive

later than the step period can be counted as lost packets [12].

The network topology and configuration influence the packet delay. Moreover, the packet delay may be different depending on the direction of traffic flow, i.e. the upstream and downstream latency may be different. WirelessHART and OpenWSN both allocate more bandwidth for upstream traffic by default, which typically results in lower latency for upstream packets compared to downstream packets.

3.2 Hardware

DecaWave produce an UWB transceiver chip which is compliant with IEEE 802.15.4-2011, called the DW1000. The DW1000 is designed for communication according to IEEE 802.15.4-2011 and for precise ranging with an accuracy of ±10 cm. The list of features of the DW1000 is shown in Table 3.1. Figure 3.2 shows an image of the DW1000 tranceiver.

Table 3.1: Capabilities of the DecaWave DW1000 (from [14, 17]).

Feature	DW1000 Capability
Communication Range	300 m (LOS), 40 m (NLOS)
Data Rates	110 kbps, 850 kbps, 6.81 Mbps
Frequency Range	3244.8 MHz to 7030.4 MHz
UWB Channels	1, 2, 3, 4, 5, 7
Channel Bandwidths	499.2 MHz, 1081.6 MHz, 1331.2 MHz

The EVB1000 is an evaluation board provided by DecaWave which features a DW1000 transceiver, an external omnidirectional UWB antenna, an ARM Cortex-M3 microcontroller, and a 2-line 16-character LCD. The ARM Cortex-M3 microcontroller features 256 KB of internal flash memory and 64 KB of random accesss memory. It can be programmed with user application code to control the DW1000 via a dedicated serial peripheral interface (SPI) bus. External communication to the ARM microcontroller is possible via a universal serial bus (USB) connector on the EVB1000.

The EVB1000 is shown in Figure 3.3. Six EVB1000 boards were used for this thesis.

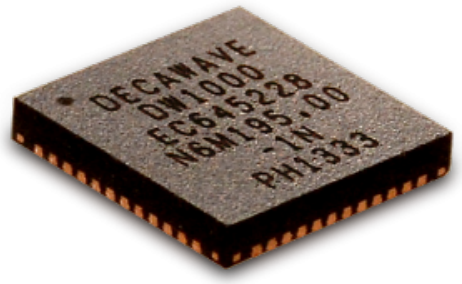


Figure 3.2: DecaWave DW1000 transceiver (from [17]).

Figure 3.3: DecaWave EVB1000 (from [14]).

3.3 OpenWSN

OpenWSN [48] is an open-source project to implement a full protocol stack for low-rate WSNs, written in the C programming language. The stack is based on the IEEE 802.15.4-2006 2.435 GHz offset quadrature phase-shift keying (O-QPSK) physical layer and the IEEE 802.15.4e-2012 time slotted channel hopping (TSCH) MAC layer. The full protocol stack provided by OpenWSN is shown in Figure 3.4. The network layer of OpenWSN implements the Internet protocol version 6 (IPv6), and routes messages using the routing protocol for low-power and lossy networks (RPL) as defined by the Internet engineering task force (IETF) in RFC 6550 [47]. RPL organizes the routing paths to form a directed acyclic graph (DAG). One node in the network is designated the *DAG root*, which is the node at which all paths terminate. This network organization is well suited to WSNs found in industrial applications, where one node in the network (i.e. the DAG root) acts as an access point to connect the wireless network to a wired backbone network. A detailed description of the protocol stack elements shown in Figure 3.4 can be found in [37].

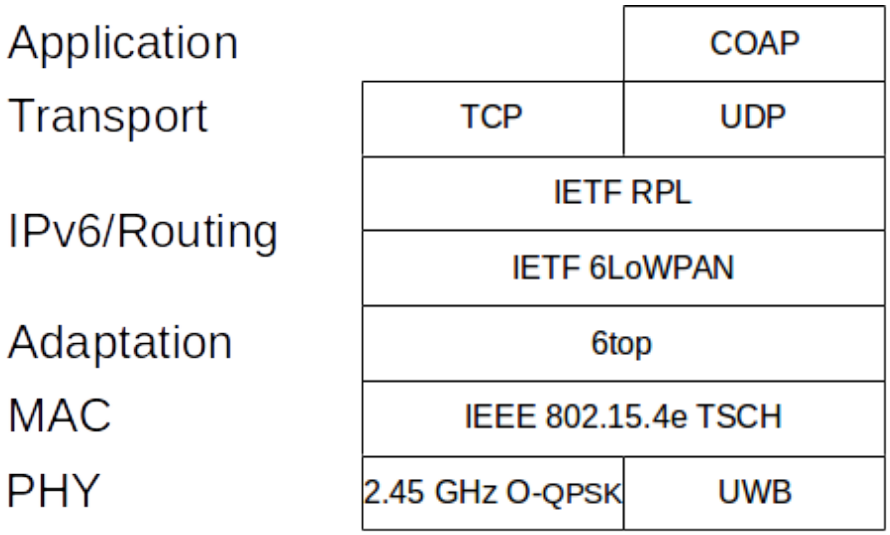


Figure 3.4: The OpenWSN protocol stack.

The TSCH schedule determines the time slot and channel used for communication with neighboring motes. Each schedule entry encapsulates the slot type, the MAC address of the neighbor mote, and the slot offset and channel offset used for communicating with the neighboring mote. The type is set to one of:

- CELLTYPE_RX indicates that the mote receives a packet from a neighboring mote in this slot.
- CELLTYPE_TX indicates that the mote transmits a packet in this slot.
- CELLTYPE_TXRX indicates that the slot is used for both transmitting and receiving packets. This type is used for shared slots.
- CELLTYPE_SERIALRX indicates that the slot is used for receiving data from its serial link, instead of wireless communication.

An example of a network schedule is shown in Table 3.2.

Table 3.2: An example TSCH schedule.

Channel offset	Slot Offset 0	1	2	3	4
0	shared				
1		$A \rightarrow C$			$C \rightarrow A$
2				$B \rightarrow D$	
3			$D \rightarrow A$		
4					
5					
6		$B \rightarrow D$	$B \rightarrow C$		
7					

3.3.1 Architecture

OpenWSN consists of two parts: the firmware which implements the protocol stack shown in Figure 3.4, and the OpenVisualizer software which runs on a host computer. OpenVisualizer facilitates the connection between the Internet and the OpenWSN mesh network via a serial connection to the DAG root mote. The DAG root mote transfers received packets over the serial connection to OpenVisualizer where the 6LoWPAN headers are converted to full IPv6 headers. The packet is then transferred to a virtual tunnel where the packet is delivered to the destination application via a standard socket.

Each mote in the mesh network has its own IPv6 address, which permits applications running on the host computer to send messages to specific motes via OpenVisualizer. Messages sent to a mote in the mesh network are converted from full IPv6 packets to 6LoWPAN packets by OpenVisualizer before being sent to the DAG root mote via the serial link where the packet is then routed to the destination mote. This architecture is shown in Figure 3.5.

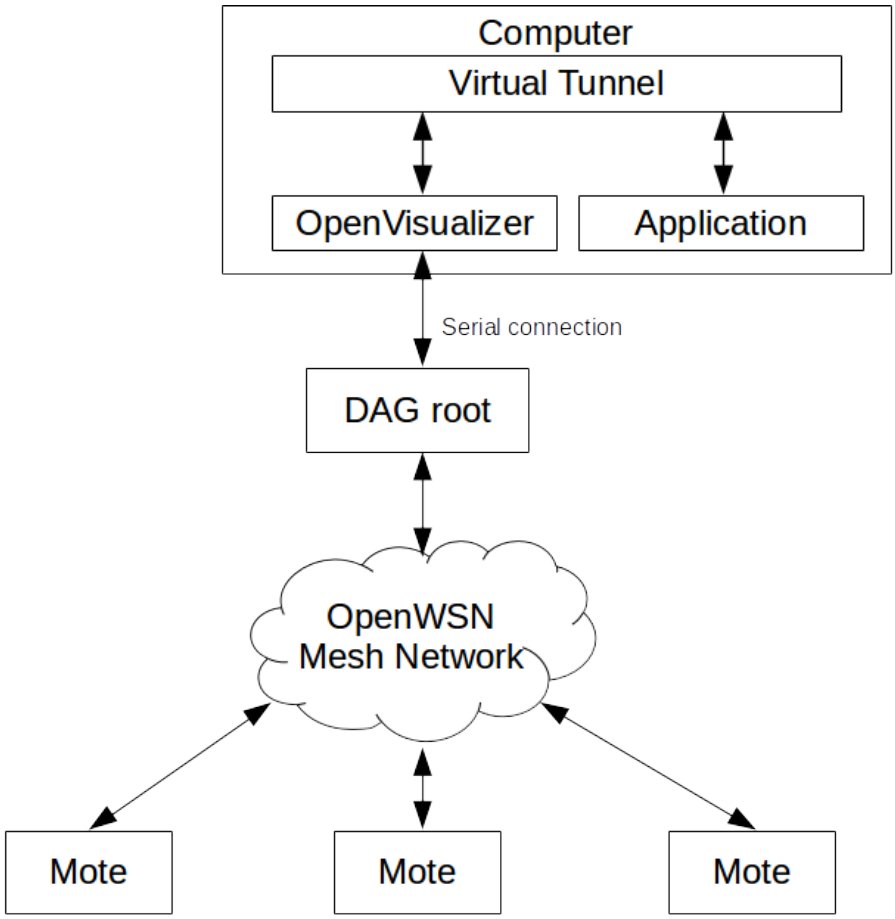


Figure 3.5: OpenWSN network architecture.

3.3.2 Adaptation to UWB

OpenWSN supports several hardware platforms, such as IoT Lab M3, OpenMote, and TelosB. However, all current platforms are based on the 2.45 GHz O-QPSK physical layer defined in IEEE 802.5.4. In this thesis we have ported OpenWSN to the UWB physical layer, running on the ARM Cortex-M3 microcontroller of the DecaWave EVB1000 hardware. This section describes the details of how UWB is used with OpenWSN.

3.3.2.1 Channel Configuration

OpenWSN uses TSCH over multiple physical channels. For the 2.45 GHz O-QPSK physical layer, channels 11 – 26 are used for channel hopping. The DecaWave

DW1000 UWB radio, does not support sixteen different channels (only channels 1 – 5 and 7 are supported). UWB does permit multiple users to use the same channel at the same time, as long as different preamble codes are used. The choice of preamble code is important in order to avoid interference between multiple users of the same channel.

Testing by DecaWave [15] has shown that two radios on the same channel interfere with each other only when using preamble codes with the same PRF and within a certain distance of each other. With a PRF of 64 MHz the distance where the radios interfere is between 29 m and 48 m, depending on the channel used. With a PRF of 16 MHz, the interference distance is between 57 m and 80 m. When two radios use different PRFs, DecaWave experimentally demonstrated that the radios do not interfere. This allows us to use each channel at most twice simultaneously, as long as a different PRF is used.

Channels 4 and 7 cannot be used as they have a wider bandwidth and overlap with channels 1 – 3 and 5 respectively. Their use would cause cross-channel interference on those overlapping channels. For this reason we only use the four channels: 1, 2, 3, and 5. Since each channel may be used twice (with different PRFs), this results in eight channels that can be used for TSCH in OpenWSN. The resulting channel configuration used for the UWB physical layer in OpenWSN is shown in Table 3.3. The data rate is set to 850 kbps to provide the best reliability without exceding the duration of a time slot. In OpenWSN the time slot duration is configured to 15 ms. Within each time slot up to two packets can be exchanged: a data packet and an acknowledgement packet. The time taken to transmit both packets (data and acknowledgement) must not exceed the 15 ms time slot duration. The longest possible duration for a packet sent at 110 kbps and a preamble length of 1024 symbols is 10.7 ms, which is too long to permit two packets to be sent in a 15 ms time slot. Therefore, the 110 kbps data rate cannot be used with OpenWSN. Using the

Table 3.3: OpenWSN channel configuration for the UWB physical layer.

OpenWSN Channel	UWB Channel	Preamble Length (symbols)	Preamble Code	PRF (MHz)	Data Rate (kbps)
0	1	1024	9	64	850
1	2	1024	10	64	850
2	3	1024	11	64	850
3	5	1024	12	64	850
4	1	1024	1	16	850
5	2	1024	3	16	850
6	3	1024	5	16	850
7	5	1024	4	16	850

configuration defined in Table 3.3 the longest possible duration of a packet is 2.23 ms, which is well within the 15 ms duration of a time slot.

3.3.2.2 Channel Hopping Sequence

The channel hopping sequence used by OpenWSN is designed for 16 channels. For UWB there are only 8 channels available, so it is necessary to adapt OpenWSN's channel hopping sequence for half the number of channels. The OpenWSN channel hopping sequence for UWB is generated using the default hopping sequence defined in the IEEE 802.15.4 standard. The same algorithm is used to produce the channel hopping sequence for 8 channels.

The default hopping sequence is based on the output of a 9-bit linear feedback shift register (LFSR) using the polynomial $x^9 + x^5 + 1$ and an initial value of 255. The output of the LFSR is modulo *macHoppingSequenceLength*, which is 8 for the UWB physical layer. The algorithm to generate the IEEE 802.15.4 default channel hopping sequence is as follows:

1. Set the SHUFFLE array to the first *macHoppingSequenceLength* outputs of the 9-bit LFSR. Each LFSR output is modulo *macHoppingSequenceLength*.

2. Set the CHANNELS array to the monotonically increasing sequence of chan-

nels. The values in the "OpenWSN channels" column shown in Table 3.3 is used for the contents of the CHANNELS array.

3. Execute the steps shown in Figure 3.6.

4. The resulting contents of the CHANNELS array are used as the channel hopping sequence.

The resulting channel hopping sequence is: 5 1 4 0 3 2 6 7.

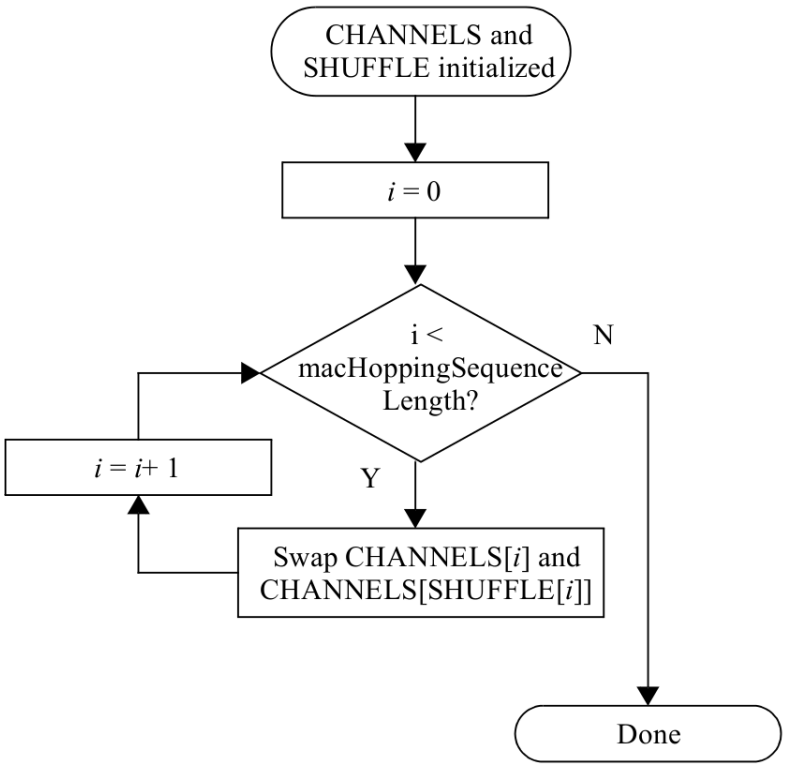


Figure 3.6: IEEE 802.15.4 default hopping sequence algorithm (from [3]).

3.3.2.3 RSSI Estimation

OpenWSN uses the received signal strength indication (RSSI) for packets received from each neighbor to determine the best neighboring motes for routing and time synchronization. The RSSI for UWB is estimated using the technique provided by DecaWave in [17] based on the number of received preamble symbols N_p and the

channel impulse response power C. Both values are provided by the DW1000 in its register set. This DecaWave calculation for the estimated RSSI in dBm is shown in Equation 3.6.

$$RSSI = 10 \times \log_{10}\left(\frac{C \times 2^{17}}{N_p^2}\right) - A \tag{3.6}$$

The value for A depends on the PRF in use. $A = 113.77$ for a PRF of 16 MHz and $A = 121.74$ for a PRF of 64 MHz.

The calculated RSSI is accurate when the actual received power is lower than -90 dBm. As the actual received power exceeds -90 dBm, the calculation shown in Equation 3.6 underestimates the receive power. The relationship between the actual received power and calculated RSSI is shown in Figure 3.7.

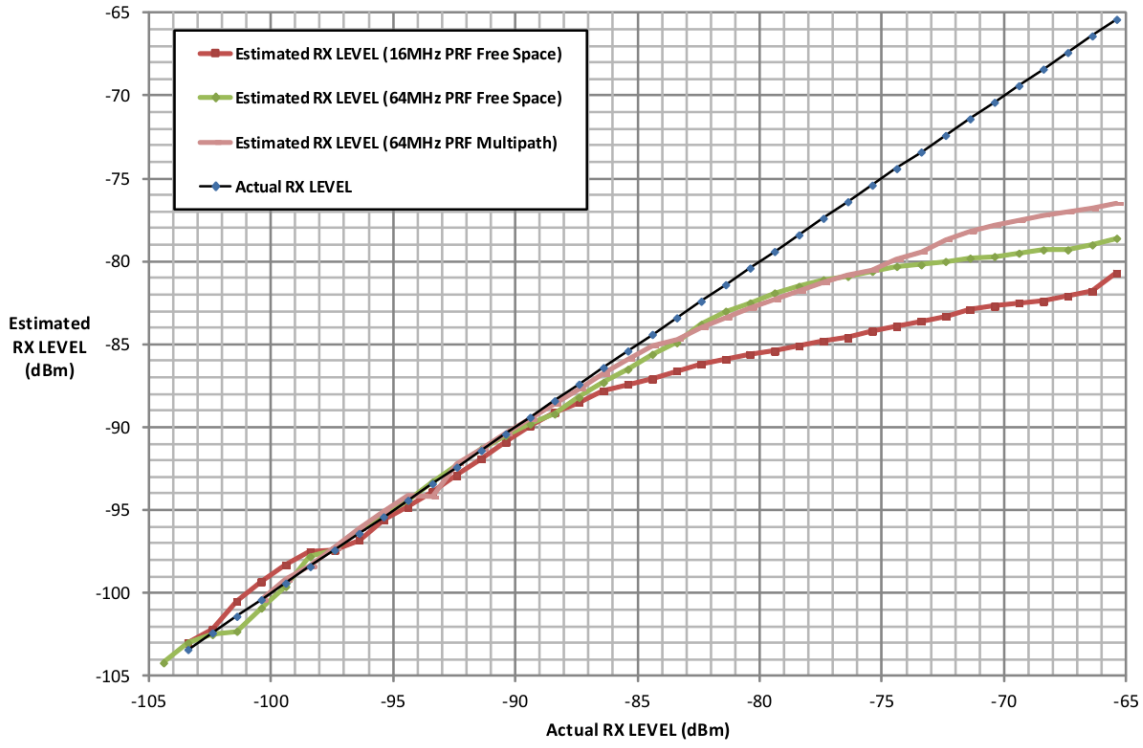


Figure 3.7: The relationship between the estimated received power and actual received power (from [17]).

Chapter 4

Experimental Comparison of Ultra-Wideband and WirelessHART

4.1 Testing Environment

The performance of UWB and WirelessHART is evaluated at two locations on the University of New Brunswick (UNB) campus: an office/laboratory at the UNB ITC building, and an industrial environment at the UNB heating plant.

4.1.1 UNB ITC Building

The UNB ITC building is the main building for the Faculty of Computer Science (FCS) at UNB. The building has three floors: a basement level one floor below ground level, one floor at ground level, and one floor above ground level. The radios in the UNB ITC Building are placed on B level, which is one floor below ground level. The layout of B level, and the position of each radio is shown in Figure 4.1. The position of each radio is marked with an X with a label to denote the number of the

transmitter or receiver, e.g. "R3" indicates receiver number 3, and "T1" indicates the position of the transmitter. One UWB radio and one WirelessHART radio are placed at each radio position.

Each receiver is placed in a non-line-of-sight (NLOS) position w.r.t. the transmitter, i.e. there are one or more obstructions in the straight line path between the transmitter and receiver antennas such as walls, equipment, and furniture. Table 4.1 shows the straight line Euclidean distance of each receiver to the transmitter, as calculated from measurements along cartesian axes with a measuring tape. The antenna height is measured from the plant floor to the base of the antenna.

Table 4.1: Straight-line distance from the transmitter (T1) to each receiver for the radio positions in the UNB ITC building.

Receiver	Antenna height (m)	Distance to transmitter (m)
R1	0.75	11.61
R2	2.43	10.52
R3	0.75	11.87
R4	2.43	12.98
R5	2.43	12.46

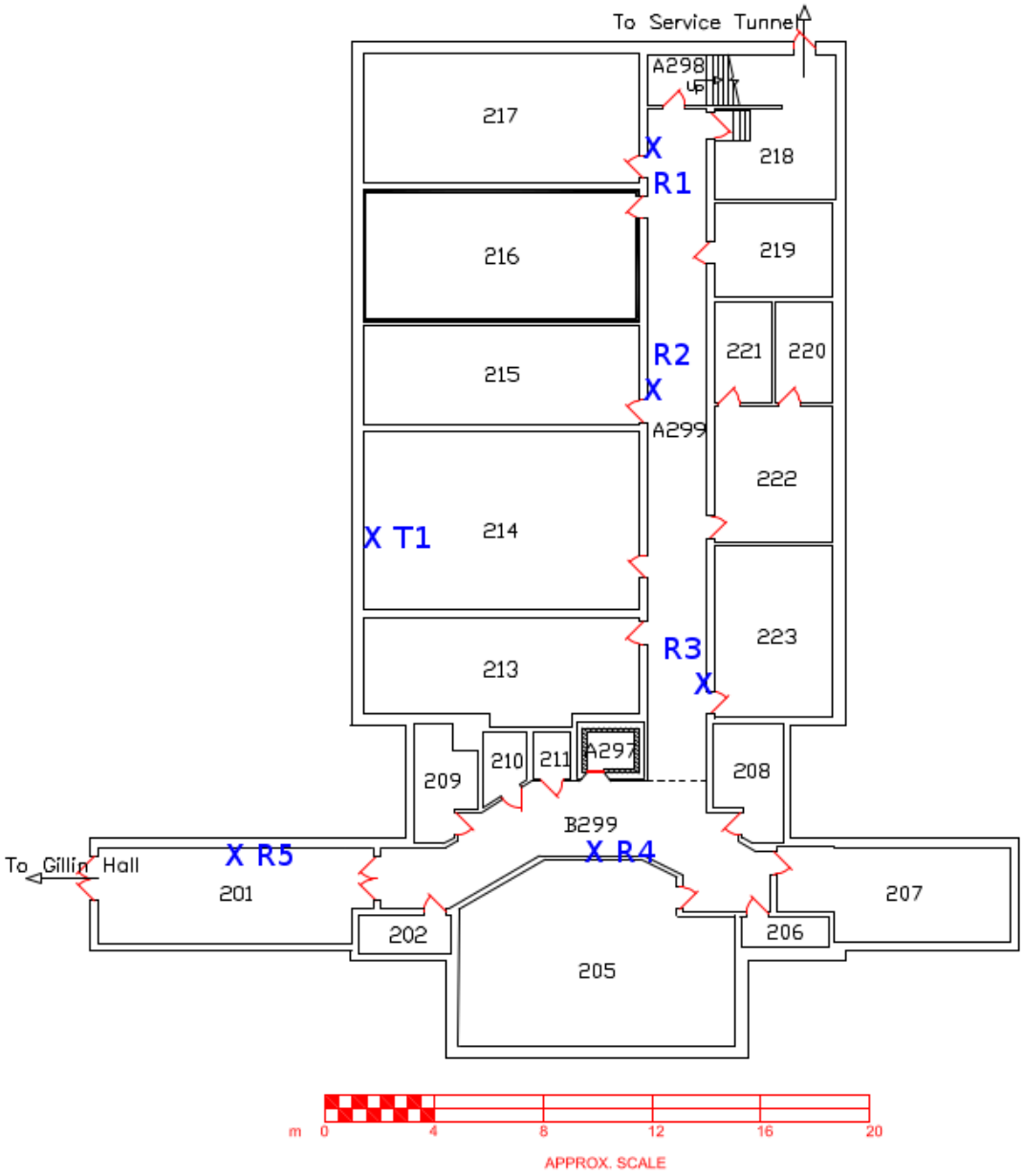
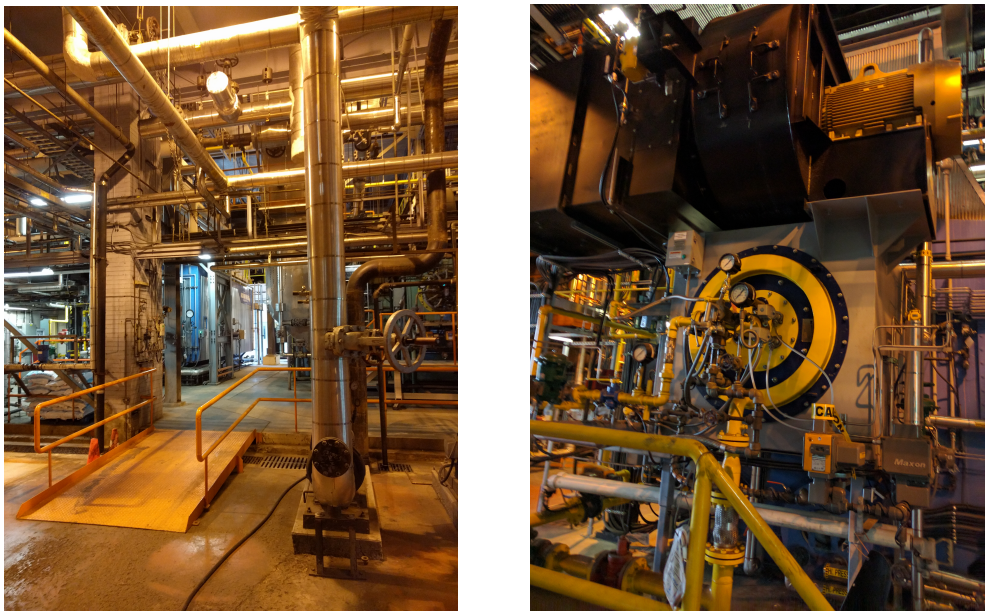


Figure 4.1: The placement of the UWB and WirelessHART radios in the UNB ITC building.

4.1.2 UNB Heating Plant

The UNB heating plant provides steam to the UNB and Saint Thomas University campuses, as well as the Doctor Everett Chalmers hospital. The heating plant contains five large industrial boilers to produce steam which are fueled by wood, oil, or natural gas. The plant also contains many pipes to carry fuel, water, chemicals, and steam. The boilers are controlled by programmable logic controllers (PLCs) which are connected to the control room via a wired network. The interior of the UNB heating plant and one of the steam boilers is shown in Figure 4.2. Images of the motes' placement in the heating plant are presented in Appendix A.



(a) (b)

Figure 4.2: (a) heating plant interior and (b) boiler #2.

The height of the plant's steel roof in the main building is 11.06 m from the plant floor. The roof is made of steel, so it may reflect radio signals within the building, aiding in the reception of NLOS signals.

Figure 4.3 shows the floor plan of the UNB heating plant with the position of the one transmitter and six receivers. The position of each radio is marked with an X

with a label to denote the number of the transmitter or receiver, e.g. "R3" indicates receiver number 3, and "T1" indicates the position of the transmitter. One UWB radio and one WirelessHART radio are placed at each radio position.

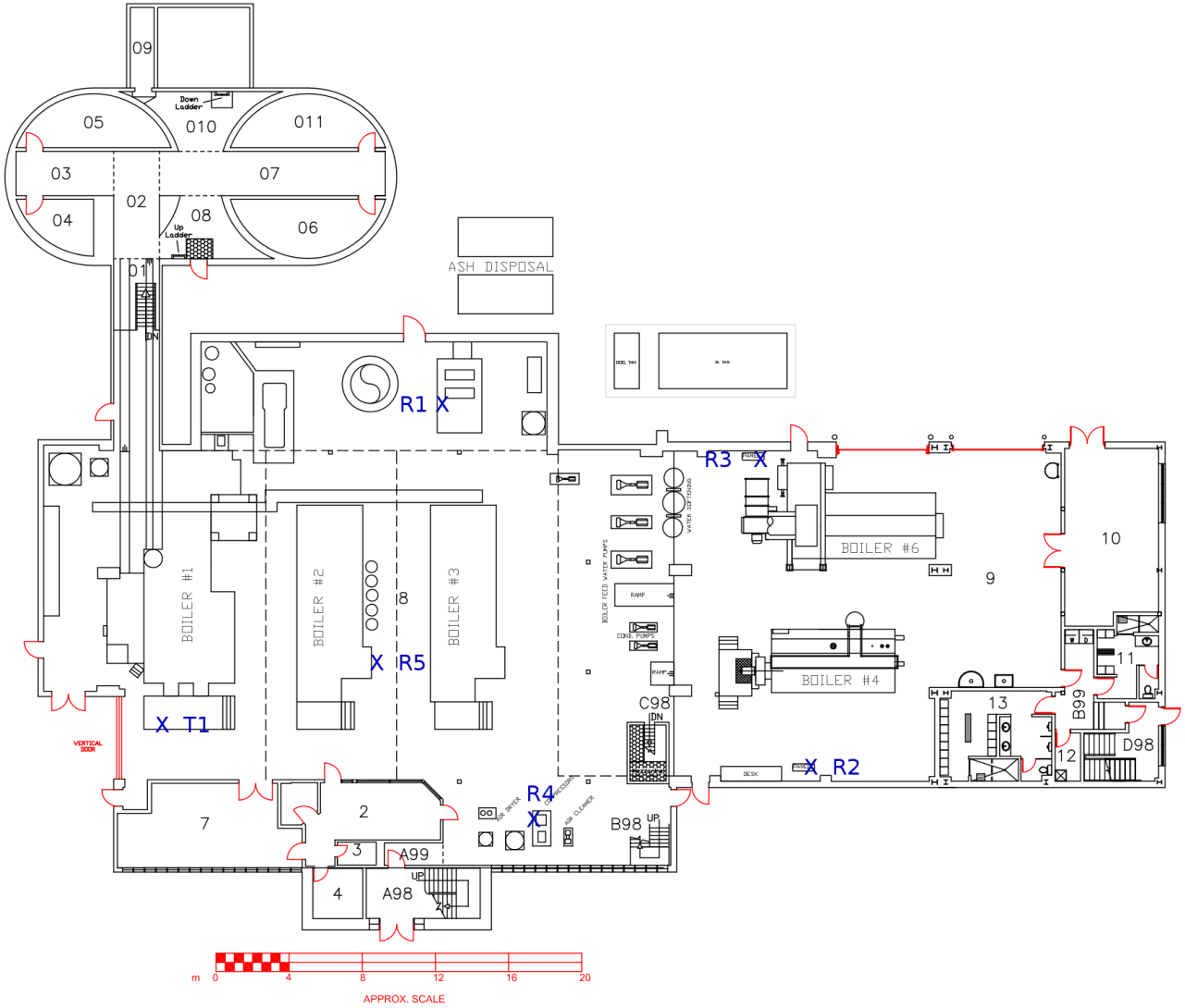


Figure 4.3: The placement of the UWB and WirelessHART radios in the UNB heating plant.

The transmitter ("T1") is placed in front of boiler #1 on top of a metal platform. The transmitter's antenna is at a height of 0.99 m from the floor.

Each receiver is placed in a NLOS position w.r.t. the transmitter, i.e. there are one or more obstructions in the straight line Euclidean path between the transmitter and receiver antennas such as boilers, machinery, and pipes. Table 4.2 shows the straight line Euclidean distance of each receiver to the transmitter, as measured using a measuring tape.

Table 4.2: Straight-line distance from the transmitter (T1) to each receiver for the radio positions in the UNB heating plant.

Receiver	Antenna height (m)	Distance to transmitter (m)
R1	1.85	24.60
R2	2.11	34.93
R3	1.95	33.33
R4	1.76	19.91
R5	2.25	14.19

The height of the roof in the area enclosing R1 is lower, at a height of 4.03 m above the floor. The area enclosing R4 is underneath the first floor walkways, which are at a height of 2.88 m above the floor.

4.1.3 Choosing Mote Positions

The radio positions in each environment were chosen so that the network performance can be tested in a difficult RF environment. All motes are placed in NLOS w.r.t the base station, and at least one mote is placed out of range w.r.t the base station in order to test the network performance over multi-hop links. Some motes in the network may have LOS to other motes in the network.

These specific positions where chosen using the UWB BER test firmware described in Section 4.2. The transmitter was placed at the position "T1" and the receiver was

moved around the environment. The positions were chosen based on the number of packets that were received at each position. Positions "R5" at the UNB ITC building and "R3" at the UNB heating plant were chosen as they were just out of range of packet reception and would require a router to be able to participate in a mesh network. The remaining positions were chosen to provide potential router mote locations in the mesh network.

4.2 UWB Bit Error Rate Measurement Procedure

As described in Section 2.2.2.1 the BER has a direct relationship with the IEC 61508 SIL. In this experimental procedure, the BER of UWB in an industrial environment is evaluated using the DecaWave DW1000 hardware described in Section 3.2.

4.2.1 Radio Configurations

The IBER is measured for each combination of UWB channel, data rate, and PRF supported by the DecaWave DW1000. This produces a total of 36 possible combinations, which are shown in Table 4.3.

All 36 tests use the same preamble length of 4096 symbols, as this gives the best possible range and performance. The duration of the preamble is approximately 4 ms.

4.2.2 Evaluation Approach

A total of six DW1000 radios are used, and are placed in the UNB heating plant as shown in Figure 4.3. One radio, denoted as T1 in Figure 4.3 is responsible for transmitting the required number of packets. The remaining five radios are placed

Table 4.3: UWB BER experiment radio configurations.

Test	UWB Channel	Center frequency (MHz)	Bandwidth (MHz)	Data Rate (kbps)	PRF (MHz)
1	1	3494.5	499.2	110	16
2	1	3494.5	499.2	110	64
3	1	3494.5	499.2	850	16
4	1	3494.5	499.2	850	64
5	1	3494.5	499.2	6810	16
6	1	3494.5	499.2	6810	64
7	2	3993.6	499.2	110	16
8	2	3993.6	499.2	110	64
9	2	3993.6	499.2	850	16
10	2	3993.6	499.2	850	64
11	2	3993.6	499.2	6810	16
12	2	3993.6	499.2	6810	64
13	3	4492.8	499.2	110	16
14	3	4492.8	499.2	110	64
15	3	4492.8	499.2	850	16
16	3	4492.8	499.2	850	64
17	3	4492.8	499.2	6810	16
18	3	4492.8	499.2	6810	64
19	4	3993.6	1331.2	110	16
20	4	3993.6	1331.2	110	64
21	4	3993.6	1331.2	850	16
22	4	3993.6	1331.2	850	64
23	4	3993.6	1331.2	6810	16
24	4	3993.6	1331.2	6810	64
25	5	6589.6	499.2	110	16
26	5	6589.6	499.2	110	64
27	5	6589.6	499.2	850	16
28	5	6589.6	499.2	850	64
29	5	6589.6	499.2	6810	16
30	5	6589.6	499.2	6810	64
31	7	6589.6	1081.6	110	16
32	7	6589.6	1081.6	110	64
33	7	6589.6	1081.6	850	16
34	7	6589.6	1081.6	850	64
35	7	6589.6	1081.6	6810	16
36	7	6589.6	1081.6	6810	64

in the other locations, and measure the BER from the packets received from the T1 radio.

For each test configuration, the IBER (see Section 3.1.1) is measured by transmitting a total of 4.6×10^7 bits and counting the number of bits that are received in error. The maximum packet size is 127 bytes, which results in a total of 45276 packets to send for each test. This data size permits measuring a BER of up to $1 \cdot 10^{-7}$ with a confidence level of 99% [43], if no packets are lost.

In order for the receiver to be able to count the number of bit errors, it must know

the correct bit pattern against which the received packet is compared. For this experiment we use a fixed bit pattern of alternating ones and zeroes (AA in hexadecimal) for all packets. The use of a fixed bit pattern avoids the need to synchronize the transmitter and receiver. The last two bytes of the packet consists of the 16-bit CRC-CCITT computed over the preceding 125 bytes. This is used to detect any residual errors that occur during the test. The resulting CRC is the two byte hexadecimal sequence 08 C8.

As mentioned in Section 2.3.2.1, the bit error probability is asymmetric for a 1 versus a 0 for a UWB PPM modulation scheme. Thus, the expected number of bit errors in a packet containing all ones may be different than for a packet containing all zeroes. The expected number of bit errors using an alternating ones and zeroes bit pattern thus represents an average for real-world packets.

4.3 Comparison between WirelessHART and UWB

The goal of this experiment is to measure and compare the performance of UWB and WirelessHART mesh networks. For the UWB network the adaptation of OpenWSN to UWB described in Chapter 3 based on the DecaWave EVB1000 is used. For WirelessHART the SmartMesh WirelessHART software development kit (SDK) is used, as described in Section 4.3.1.

The measured performance metrics for the network are: latency, packet loss, and packet acknowledgement ratio (PAR). The PAR is calculated as the number of successfully sent packets (i.e. packets that have been acknowledged) divided by the number of sent packets. The PAR of a link between two motes is influenced by both the PER and the PLR, as either the data packet or the acknowledgement packet may be lost or corrupted.

4.3.1 WirelessHART Hardware

SmartMesh WirelessHART is a product line from the Dust Networks group of Linear Systems for devices implementing the WirelessHART standard. The SmartMesh WirelessHART software development kit (SDK) is available for purchase from Linear Systems and includes the necessary equipment for research and development with WirelessHART. The SDK consists of a WirelessHART network manager and five WirelessHART motes. Figure 4.4 shows a picture of the network manager and mote hardware.

(a) (b)

Figure 4.4: SmartMesh WirelessHART (a) network manager and (b) mote.

The SmartMesh network manager acts as the gateway to the WirelessHART network. External networks can configure the network and communicate with the motes via the network manager's ethernet port. This port exposes a secure web interface on port 443 and an extensible markup language remote procedure call (XML-RPC) interface on port 4443. Linear Systems provides a free SDK software package available at [19]. The SDK provides an application programming interface (API) using the Python programming language to handle communication via the XML-RPC interface.

A host computer is connected to the network manager via the ethernet port and runs Python scripts to measure the network performance for the tests described in this

section. The Python scripts use the XML-RPC interface via the SmartMesh SDK API to send and receive packets on the WirelessHART network.

Each WirelessHART mote contains a microcontroller which runs the WirelessHART protocol stack. The mote exposes an API via a serial port to permit the mote to be controlled externally. The API is used to control the mote joining process to the WirelessHART network and to send and receive packets. For our tests, the mote is controlled via the serial port by test software running on an Arduino Due. The Arduino Due, shown in Figure 4.5, is an open-source microcontroller kit based on an ARM Cortex-M3 core. The test application runs on the Arduino Due and communicates with the WirelessHART mote through its serial port. Each WirelessHART mote is controlled by one Arduino Due board.

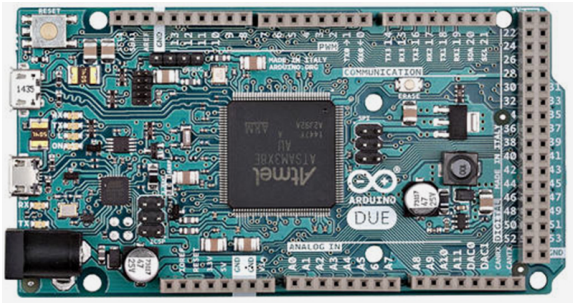


Figure 4.5: The Arduino Due microcontroller kit (from [9]).

The test setup for the WirelessHART network is shown in Figure 4.6.

4.3.2 Network Setup

The UWB and WirelessHART mesh networks are set up in the two test locations, using the radio positions shown in Figure 4.1 at the UNB ITC building, and Figure 4.3 at the UNB heating plant. One UWB mote and one WirelessHART mote are placed at each position within 10 cm of each other. The WirelessHART and UWB motes are unlikely to interfere with each other as the used frequency bands do not overlap (see Sections 2.1.1 and 3.2).

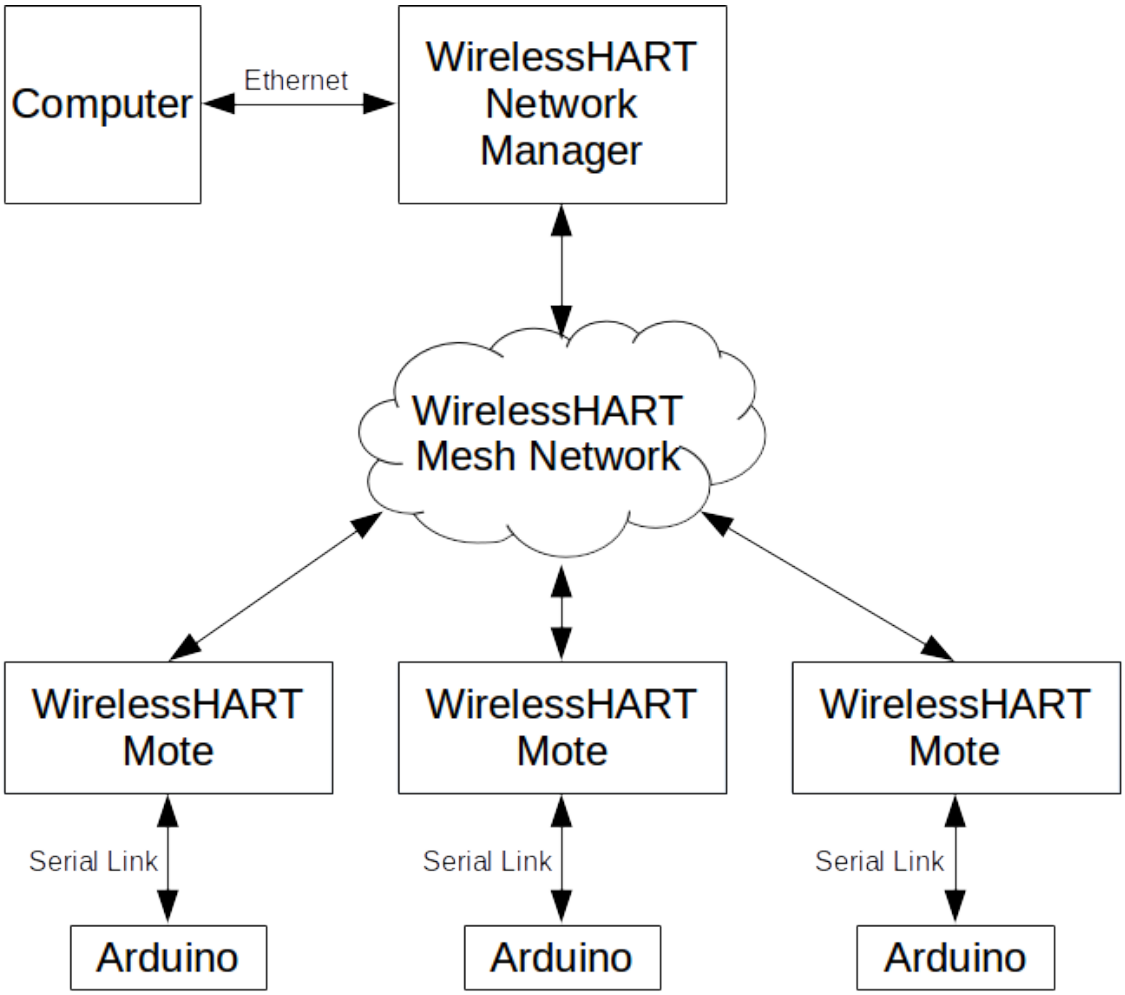


Figure 4.6: The WirelessHART test setup.

The position "T1" indicates the position of the *base station* for the network. For OpenWSN the base station is the DAG root, and for WirelessHART it is the network manager. The remaining five positions "R1" to "R5" are the remote motes which form the mesh network.

The mesh network is formed by powering on the motes and allowing them to join the network. No restrictions are placed on the network topology; the network is responsible for deciding the best network topology. The experiment does not begin until all motes have joined the network.

4.3.3 Evaluation Approach

The performance metrics are measured from two scenarios in which packets are sent through the mesh network in both upstream and downstream directions. The first scenario runs a WSN where motes publish packets at a fixed interval of 2 seconds. This test permits measuring of the upstream latency only, since the packets are flowing in only one direction. The second scenario runs a WSAN where packets are flowing in both upstream and downstream directions, which permits measuring the round trip time (RTT), as well as the upstream and downstream latency.

In both scenarios the test is controlled by the host computer connected to the OpenWSN DAG root and WirelessHART network manager. The test applications running on the motes listen to commands from the test application running on the host computer.

4.3.3.1 Upstream Scenario

During the upstream scenario each mote in the network transmits a packet to the host computer at a fixed rate of 1 packet every 2 seconds. The packet flow through the network is illustrated in Figure 4.7.

Figure 4.7 shows the point in the flow at which timestamps are captured in the OpenWSN and WirelessHART networks. These timestamps are explained in Sections 4.3.4.1 and 4.3.5.1.

The test is started from the test application by sending a command to each mote to signal the start of the test. The command contains the number of packets that should be generated by the mote. After receiving the commands the motes start generating packets at the fixed interval of 2 seconds.

The `uperiodicpush` application runs as part of the OpenWSN firmware and periodically generates packets that are sent upstream to the DAG root. The source code for the `uperiodicpush` application is provided in Appendix B.

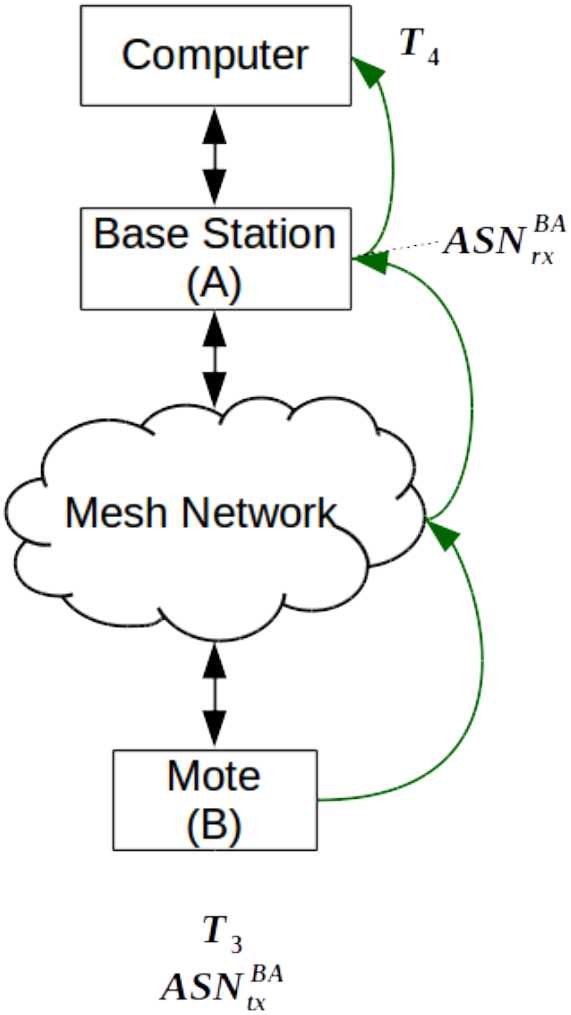


Figure 4.7: Upstream packet flow.

4.3.3.2 Round Trip Scenario

The round trip scenario measures the RTT as well as the upstream and downstream latency. Request packets are generated by the host computer and are sent to a mote which generates a reply packet to send back to the host computer. The round trip time is measured as the time elapsed between sending the request and receiving the response. The packet flow for this scenario is shown in Figure 4.8.

The `urttlatency` application runs as part of the OpenWSN firmware and sends a response packet to any request that it receives. The source code for the `urttlatency` application is provided in Appendix C.

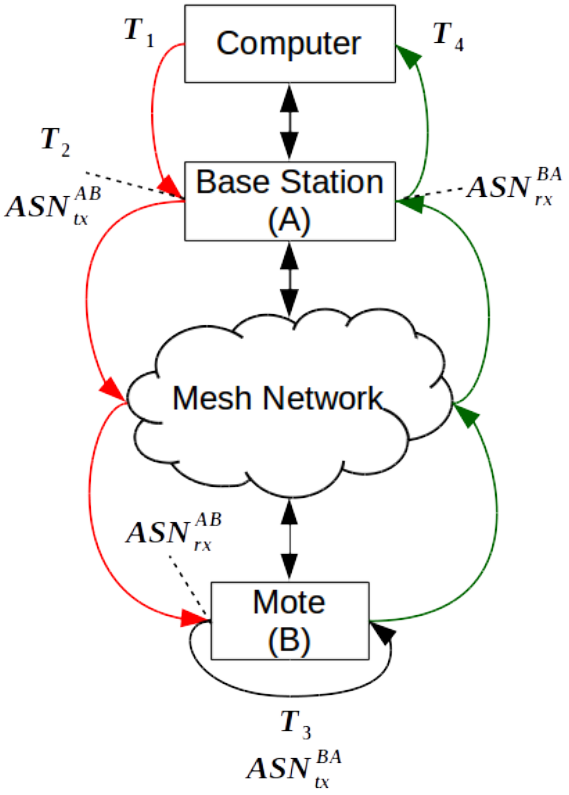


Figure 4.8: Round trip packet flow.

4.3.3.3 Communication Protocol

Typical industrial control networks only care about the most recent sensor data. In cases where packets are delayed or arrive out of order, the communication protocol should not wait for older packets to be received when more recent data is available. Most industrial communication protocols are based on the user datagram protocol (UDP), or a similar protocol. For the experiments in this thesis data is exchanged using an unreliable protocol such as UDP.

OpenWSN provides an implementation of UDP and a partial implementation of the transport control protocol (TCP). For our experiments all packets are sent over UDP. WirelessHART does not support TCP or UDP, but provides similar behaviour through *acknowledged* or *best effort* communication. Packets sent using acknowledged communication require an end-to-end acknowledgement to be sent from the receiving mote. If the network manager does not receive the acknowledgement in a

timely manner then the packet is retransmitted by the manager. In this manner, *acknowledged* communication is similar to TCP.

Packets sent using best effort communication do not require an end-to-end acknowledgement. The manager does not retransmit the packet if it fails to reach its destination.

Although all packets are transmitted using a UDP-like protocol which does not acknowledge packets at the transport layer, packets are acknowledged at the MAC layer per-hop. When a packet is routed to its next hop, the sender waits for an acknowledgement at the MAC layer. If no ackowledgement is received, then the MAC layer will attempt to retransmit the packet at the next available time slot for up to 4 attempts in total. This occurs for both WirelessHART and OpenWSN.

4.3.3.4 Packet Loss

The PLR is calculated by the test applications running on the host computer. It is calculated as the number of packets received divided by the expected number of packets. This calculation is the same for both OpenWSN and WirelessHART networks.

Each packet generated by the test applications contains a unique sequence number which increases monotonically for each packet. The sequence number is used to detect lost and duplicate packets.

4.3.4 OpenWSN Measurements

4.3.4.1 Latency

In OpenWSN, each mote is synchronized to within 1 ms of each other (see Section 2.4.2.1), but do not share a common clock for hours, minutes, and seconds. All motes are synchronized to the network ASN, a monotonically increasing counter which increments every 15 ms, i.e. the time slot duration. This permits the ASN to

be used as a timestamp with a precision of 15 ms.

The upstream latency is calculated from the two ASNs shown in Figure 4.7:

- ASN_{tx}^{BA} is the ASN in which the periodic response packet is generated by the mote (B) and queued for transmission to the DAG root (A).
- ASN_{rx}^{BA} is the ASN in which the periodic response packet is received at the base station. The DAG root mote in OpenWSN captures this ASN and forwards the value to OpenVisualizer, where the upstream latency is calculated.

The upstream latency in milliseconds T_{up} is thus calculated using the number of 15 ms time slots between the generation of the packet at the mote and the packet received at the DAG root, as shown in Equation 4.1. The timestamp measurement is accurate to within 1 time slot, so the resolution of T_{up} for OpenWSN is ± 15 ms.

$$T_{up} = (ASN_{rx}^{BA} - ASN_{tx}^{BA}) \times 15 \tag{4.1}$$

For the round trip scenario, the upstream latency is calculated the same way as shown in Equation 4.1. The RTT is calculated by the test application using the following two timestamps:

- T_1 is the time at which the round trip request packet is generated in the test application and sent to OpenVisualizer for transmission into the mesh network.
- T_4 is the time at which the round trip reply packet is received at the host computer from the target mote.

The timestamps for T_1 and T_4 are captured from the host computer's local clock, and are measured in milliseconds since midnight on January 1, 1970. The round trip time T_{RTT} is calculated from T_1 and T_4 as shown in Equation 4.2

$$T_{RTT} = T_4 - T_1 \tag{4.2}$$

ASN_{tx}^{AB} and ASN_{rx}^{AB} denote the ASN in which the request message is transmitted from the DAG root and the ASN in which it is received at the destination mote respectively. Computing the downstream latency would be a simple subtraction between these two ASNs. OpenWSN, however, does not return the value ASN_{tx}^{AB} to OpenVisualizer which prevents its use in calculating the downstream latency. The downstream latency is therefore estimated from T_{RTT} and T_{up} as shown in Equation 4.3.

$$T_{down} = T_{RTT} - T_{up} \tag{4.3}$$

The value for T_{down} only provides an estimate for the downstream latency. The measurement includes queueing delays at the base station, as well as delays in sending the packet from OpenVisualizer to the DAG root mote.
For both upstream and round trip scenarios the latency information is captured and logged by OpenVisualizer. The OpenVisualiser source code to capture the ASN information is provided in Appendix D.

4.3.4.2 Packet Acknowledgement Ratio

Each mote in OpenWSN maintains a list of its neighboring motes that are within its radio range. The neighbors list also contains the following information about each neighboring mote:

- N_{rx} records the number of packets successfully received from the neighboring mote.
- N_{tx} records the number of packets that have been transmitted to the neighboring mote.
- N_{txAck} records the number of packets that have been acknowledged by the neighboring mote, indicating successful transmission.

The PAR to the neighboring mote is calculated using N_{tx} and N_{txAck} as shown in Equation 4.4.

$$PAR = \frac{N_{txAck}}{N_{tx}} \tag{4.4}$$

The neighbor statistics are requested by the host computer at the end of each test via several requests to the `unbrinfo` and `uscheduleinfo` applications running on the mote. The source code for the `unbrinfo` and `uscheduleinfo` applications are provided in Appendix E and Appendix F respectively.

4.3.5 WirelessHART Measurements

4.3.5.1 Latency

Motes in a WirelessHART network are synchronized to within 1 ms of the network manager by a mechanism similar to the IEEE 802.15.4 TSCH MAC layer used in OpenWSN. A key difference with OpenWSN is that the WirelessHART network manager distributes its local clock time to each mote as the date and time since midnight on January 1 1970. During testing with the SmartMesh WirelessHART motes, the time difference between the motes and network manager was within 10 ms.

Applications connected to the network manager via the XML-RPC interface are notified of various network events, such as: packet sent, packet received, and transport timeouts. Each notification has an associated timestamp using the network manager's local clock, which we use to measure the latency. For the WirelessHART round-trip tests we use four timestamps as shown in Figures 4.7 and 4.8:

- T_1 is the timestamp at which the request packet is generated in the test application. This timestamp is obtained by querying the network manager's local clock via the XML-RPC API.

- T_2 is the time at which the network manger sends the packet to the network. The network manager sends a notification containing this timestamp to the test application running on the host computer when the packet is sent.

- T_3 is the time at which the mote received the request and generated the response packet. The WirelessHART mote automatically captures this information and sends it to the network manager. The network manager reports this timestamp in the packet notification sent to the test applications `rtt_test.py` and `upstream_test.py` when the packet is received at the network manager. The test applications are running on the computer as shown in Figures 4.7 and 4.8.

- T_4 is the time at which the response packet is received at the test applications `rtt_test.py` and `upstream_test.py` running on the host computer. The test applications are running on the computer as shown in Figures 4.7 and 4.8. The T_4 timestamp is obtained by querying the network manager's local clock via the XML-RPC API.

The source code for the test applications `rtt_test.py` and `upstream_test.py` are provided in Appendix G and Appendix H respectively.
For the upstream tests only the timestamps T_3 and T_4 are available, since the mote is generating packets at a fixed period. The upstream latency is calculated using these two timestamps as shown in Equation 4.5.

$$T_{up} = T_4 - T_3 \tag{4.5}$$

For the round trip tests the T_1 and T_2 timestamps are also available. The RTT is calculated as the difference between the time at which the request was sent and the time at which the reply was received in the test application. This is shown in Equation 4.6.

$$T_{RTT} = T_4 - T_1 \tag{4.6}$$

The downstream latency is calculated as the time between the packet being sent to the network using the network manager's "packet sent" notification and the time at which the mote received the request and generated the reply. This is shown in Equation 4.7.

$$T_{down} = T_3 - T_2 \tag{4.7}$$

Packets sent to the network manager are not always sent to the network immediately, but are queued at the network manager for some time. We can calculate the queuing delay as shown in Equation 4.8.

$$T_{queue} = T_2 - T_1 \tag{4.8}$$

4.3.5.2 Packet Acknowledgement Ratio

The PAR statistic in WirelessHART is known as *link stability*. The WirelessHART network manager builds the network topology by assigning links between the various motes in the network. Information about these links are periodically reported to the network manager in "health reports" sent by the WirelessHART motes. The health reports contain information about the PAR for each of the mote's links to neighboring motes, and uses the same calculation as OpenWSN as shown in Equation 4.4.

The PAR information is only available for the links that are currently assigned as in-use in the network. The PAR for each link is read via the XML-RPC interface at the end of each test. Furthermore, the network manager also calculates the overall network PAR based on all links in the network.

The network manager reports the link PAR as a percentage. It does not provide

information about the number of packets sent over each link.

Chapter 5

Experimental Results

This chapter presents the results from the experiments described in Chapter 4, performed at the UNB heating plant and UNB ITC building. The first experiment measures the BER of UWB communication using the DecaWave EVB1000 hardware. The second experiment measures the performance of WirelessHART, and OpenWSN using the UWB physical layer.

The experiments were performed over five days:

- January 30, 2017: UWB BER measurements at the UNB heating plant.
- February 17, 2017: UWB BER measurements at the UNB ITC building.
- February 20, 2017: Upstream test scenario with OpenWSN and WirelessHART networks at the UNB ITC building.
- February 21, 2017: Round trip test scenario with the OpenWSN and WirelessHART networks at the UNB ITC building.
- February 22, 2017: Both test scenarios with the OpenWSN and WirelessHART networks at the UNB heating plant.

During the round trip test for mote R5 at the UNB heating plant on February 22, 2017 the mote was moved by approximatly 1 m in the direction of the control room.

The mote was moved after approximately 100 samples had been obtained. This move was necessary in order to avoid damage to the mote from welding that was occurring on the boiler next to the mote. This change only affects the round trip measurements for mote R5. All other tests were performed with the mote at its original position.

5.1 RF Measurements

RF measurements were taken before starting the tests to measure the background RF energy occurring in the test environments for the frequencies used by UWB and WirelessHART. For UWB the measured frequencies are from 3 GHz to 7 GHz. For WirelessHART the measured frequencies are from 2.4 GHz to 2.483 GHz.

The spectrum analyzer used was a SPECTRAN HF-6080 handheld spectrum analyzer made by Aaronia AG, with the serial number 01448. The antenna used was a HyperLOG 6080 antenna, also made by Aaronia AG, with the serial number 01152. The sensitivity of the spectrum analyzer is -120 dBm.

The measurements were performed using a bandwidth of 1 MHz with a sample time of 30,000 ms. The measurement period was 30 minutes in order to obtain at least 20 full sweeps of the measured frequency range for averaging. None of the UWB or WirelessHART devices are powered on for the duration of the scan.

Two values were recorded for each measurement frequency:

- *Max. hold* records the peak value seen from all measurements taken at each frequency.
- *Average* records the average power over the last 20 samples at each frequency.

Figures 5.1 and 5.2 show the output from the spectrum analyzer at the UNB ITC building on Feburary 20, 2017 for the frequency bands used by UWB and WirelessHART respectively. The RF measurements for all other testing days are shown in Appendix I.

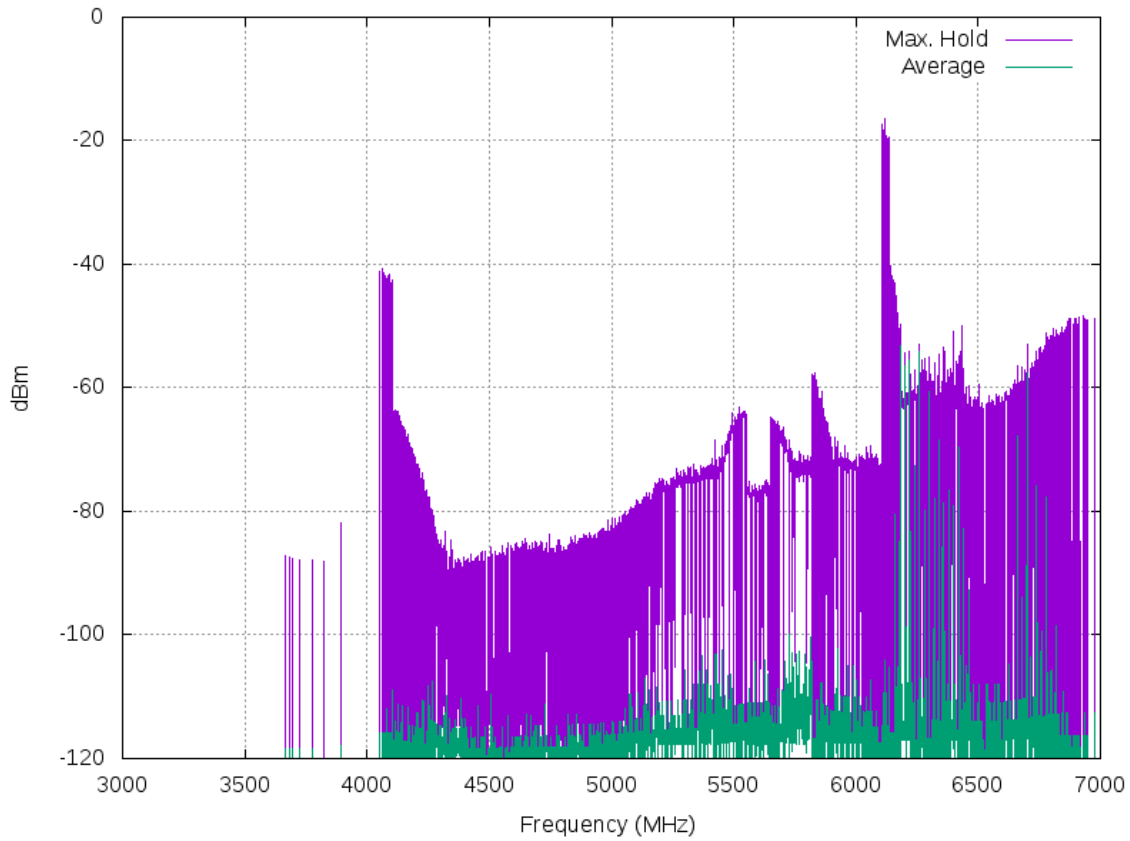


Figure 5.1: Spectrum analysis at position T1 in the UNB ITC building on February 20, 2017 from 3 GHz to 7 GHz.

The UWB spectrum measurements show two strong peaks at the frequencies 4.064 GHz and 6.105 GHz with a peak power of -40.8 and -17.5 dBm respectively. The average power at the peaks is very low, at -115.9 dBm and -114.6 dBm respectively. The average power across most of the frequency range is below -100 dBm. However, from 6.18 GHz to 6.74 GHz there are several peaks in the average power at several narrowband frequencies with the strongest average power of -53.13 dBm at 6.18 GHz. These average power peaks overlap the frequency range for UWB channels 5 and 7 which are used by the DW1000.

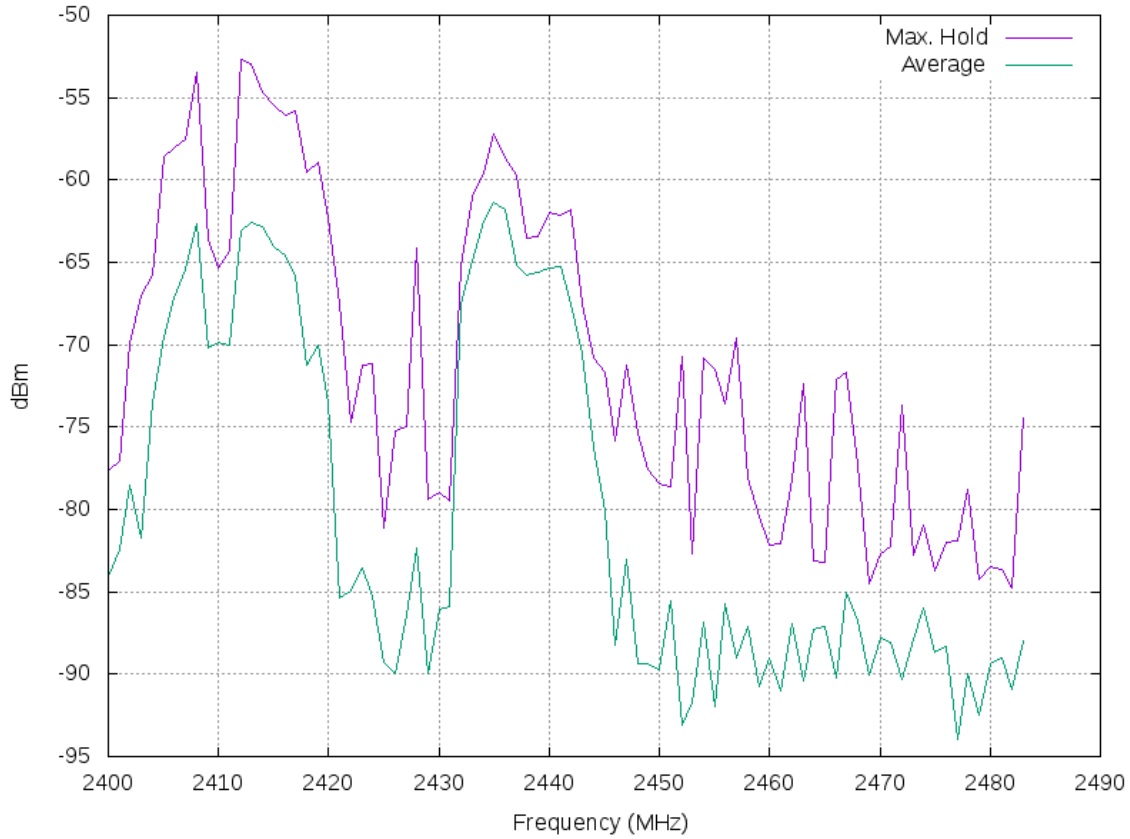


Figure 5.2: Spectrum analysis at position T1 in the UNB ITC building on February 20, 2017 from 2.4 GHz to 2.483 GHz.

In the 2.4 GHz ISM band the average power is generally between -85 dBm to -90 dBm with two peaks at 2.437 GHz and 2.412 GHz. These peaks are caused by the building's WiFi network access points which are operating on WiFi channels 1 and 6.

5.2 UWB BER Measurements

This section presents the results of the BER measurements for UWB for each of the 36 test configurations described in Section 4.2.1.

5.2.1 UNB ITC Building

5.2.1.1 Packet Loss Ratio

Not every transmitted packet was received by all receivers. Table 5.1 shows the percentage of packets that were not received for each radio position, i.e. the percentage of packets lost.

Table 5.1: Measured PLR at each radio position at the UNB ITC building.

Test	Channel	Data Rate (kbps)	PRF (MHz)	R1	R2	R3	R4	R5
1	1	110	16	1.526%	0.002%	0.000%	12.704%	99.993%
2	1	110	64	1.933%	0.002%	0.002%	17.024%	100.000%
3	1	850	16	4.605%	0.002%	0.000%	38.270%	100.000%
4	1	850	64	3.037%	0.002%	0.000%	31.204%	100.000%
5	1	6810	16	97.681%	0.044%	0.002%	99.850%	100.000%
6	1	6810	64	91.143%	0.027%	0.004%	99.867%	100.000%
7	2	110	16	10.449%	0.013%	0.013%	12.751%	99.991%
8	2	110	64	12.371%	0.007%	0.007%	13.380%	99.998%
9	2	850	16	26.869%	0.015%	0.057%	53.790%	100.000%
10	2	850	64	25.084%	0.009%	0.011%	45.386%	100.000%
11	2	6810	16	99.909%	0.121%	0.570%	99.837%	100.000%
12	2	6810	64	99.896%	0.093%	0.044%	99.841%	100.000%
13	3	110	16	57.622%	0.004%	0.000%	99.885%	100.000%
14	3	110	64	8.106%	0.007%	0.024%	51.422%	99.998%
15	3	850	16	76.652%	0.000%	0.011%	100.000%	100.000%
16	3	850	64	13.027%	0.002%	0.002%	94.483%	100.000%
17	3	6810	16	99.890%	0.038%	0.004%	100.000%	100.000%
18	3	6810	64	99.949%	0.022%	0.060%	99.947%	100.000%
19	4	110	16	3.359%	0.064%	0.051%	9.844%	99.980%
20	4	110	64	2.644%	0.044%	0.077%	4.579%	99.996%
21	4	850	16	10.745%	0.015%	0.020%	21.208%	100.000%
22	4	850	64	4.367%	0.009%	0.018%	18.811%	100.000%
23	4	6810	16	86.819%	0.042%	0.088%	99.909%	100.000%
24	4	6810	64	52.756%	0.055%	0.093%	99.969%	100.000%
25	5	110	16	5.402%	0.015%	0.007%	99.929%	99.998%
26	5	110	64	5.237%	0.007%	0.011%	98.750%	99.993%
27	5	850	16	13.303%	0.002%	0.013%	100.000%	100.000%
28	5	850	64	7.560%	0.000%	0.007%	99.989%	100.000%
29	5	6810	16	99.940%	0.049%	0.042%	100.000%	100.000%
30	5	6810	64	99.945%	0.007%	0.013%	100.000%	100.000%
31	7	110	16	14.613%	0.077%	0.011%	99.947%	99.982%
32	7	110	64	9.098%	0.102%	0.102%	99.885%	99.978%
33	7	850	16	31.496%	0.004%	0.013%	100.000%	100.000%
34	7	850	64	9.882%	0.011%	0.029%	99.996%	100.000%
35	7	6810	16	99.892%	0.022%	0.060%	100.000%	100.000%
36	7	6810	64	86.357%	0.029%	0.139%	100.000%	100.000%

Motes at positions R2 and R3 show very good packet reception, with less than 1% of packets lost for all test configurations. The motes as positions R1 and R4 show a

varying amount of packet loss, depending on all test factors. The mote at position R5 received less than 0.03% of packets at a data rate of 110 kbps. R5 was not able to receive packets at any other data rate.

5.2.1.2 Packet Error Ratio

The PER is calculated as the number of packets received containing one or more bit errors divided by the total number of packets received. Table 5.2 shows the calculated PER for each radio position. At some receiver positions no packets were received during the test, i.e. 100% of packets were lost. In this case, the PER cannot be measured and the corresponding cell in Table 5.2 is empty.

5.2.1.3 Bit Error Ratio

The IBER is calculated as the number of incorrect bits divided by the total number of bits received. Table 5.3 shows the calculated IBER at each radio position. At some receiver positions no packets were received during the test, i.e. 100% of packets were lost. In this case the IBER cannot be measured and the corresponding cell in Table 5.3 is empty.

Table 5.2: Measured PER at each radio position at the UNB ITC building.

Test	Channel	Data Rate (kbps)	PRF (MHz)	R1	R2	R3	R4	R5
1	1	110	16	$2.24 \cdot 10^{-5}$	0.00	0.00	$1.01 \cdot 10^{-4}$	1.00
2	1	110	64	0.00	0.00	0.00	$2.66 \cdot 10^{-5}$	
3	1	850	16	$1.39 \cdot 10^{-4}$	0.00	0.00	$1.14 \cdot 10^{-3}$	
4	1	850	64	$2.28 \cdot 10^{-5}$	0.00	0.00	$8.03 \cdot 10^{-4}$	
5	1	6810	16	$2.03 \cdot 10^{-1}$	$2.21 \cdot 10^{-5}$	0.00	$6.32 \cdot 10^{-1}$	
6	1	6810	64	$1.00 \cdot 10^{-1}$	0.00	0.00	$7.00 \cdot 10^{-1}$	
7	2	110	16	$9.87 \cdot 10^{-5}$	0.00	0.00	$2.53 \cdot 10^{-5}$	1.00
8	2	110	64	$5.04 \cdot 10^{-5}$	0.00	0.00	$2.55 \cdot 10^{-5}$	1.00
9	2	850	16	$6.34 \cdot 10^{-4}$	0.00	0.00	$2.10 \cdot 10^{-3}$	
10	2	850	64	$8.84 \cdot 10^{-4}$	0.00	0.00	$1.74 \cdot 10^{-3}$	
11	2	6810	16	$5.85 \cdot 10^{-1}$	$4.42 \cdot 10^{-5}$	$1.11 \cdot 10^{-4}$	$6.22 \cdot 10^{-1}$	
12	2	6810	64	$5.32 \cdot 10^{-1}$	0.00	$2.21 \cdot 10^{-5}$	$5.97 \cdot 10^{-1}$	
13	3	110	16	$5.21 \cdot 10^{-5}$	0.00	0.00	0.00	
14	3	110	64	$2.40 \cdot 10^{-5}$	0.00	0.00	$4.55 \cdot 10^{-5}$	1.00
15	3	850	16	$3.03 \cdot 10^{-3}$	0.00	0.00		
16	3	850	64	$2.54 \cdot 10^{-4}$	0.00	0.00	$4.40 \cdot 10^{-3}$	
17	3	6810	16	$5.00 \cdot 10^{-1}$	0.00	0.00		
18	3	6810	64	$6.52 \cdot 10^{-1}$	0.00	0.00	$3.33 \cdot 10^{-1}$	
19	4	110	16	$4.57 \cdot 10^{-5}$	0.00	0.00	$2.45 \cdot 10^{-5}$	1.00
20	4	110	64	$2.27 \cdot 10^{-5}$	0.00	0.00	$2.31 \cdot 10^{-5}$	1.00
21	4	850	16	$1.48 \cdot 10^{-4}$	0.00	0.00	$3.92 \cdot 10^{-4}$	
22	4	850	64	$4.62 \cdot 10^{-5}$	0.00	0.00	$3.26 \cdot 10^{-4}$	
23	4	6810	16	$5.61 \cdot 10^{-2}$	$2.21 \cdot 10^{-5}$	$2.21 \cdot 10^{-5}$	$5.12 \cdot 10^{-1}$	
24	4	6810	64	$1.64 \cdot 10^{-2}$	0.00	$2.21 \cdot 10^{-5}$	$7.14 \cdot 10^{-1}$	
25	5	110	16	$2.33 \cdot 10^{-5}$	0.00	0.00	0.00	1.00
26	5	110	64	0.00	0.00	0.00	0.00	1.00
27	5	850	16	$2.55 \cdot 10^{-4}$	0.00	0.00		
28	5	850	64	$1.67 \cdot 10^{-4}$	0.00	0.00	0.00	
29	5	6810	16	$7.41 \cdot 10^{-1}$	0.00	0.00		
30	5	6810	64	$7.20 \cdot 10^{-1}$	0.00	0.00		
31	7	110	16	$1.03 \cdot 10^{-4}$	0.00	0.00	0.00	1.00
32	7	110	64	0.00	0.00	0.00	0.00	1.00
33	7	850	16	$9.03 \cdot 10^{-4}$	0.00	0.00		
34	7	850	64	$2.45 \cdot 10^{-4}$	0.00	0.00	0.00	
35	7	6810	16	$6.53 \cdot 10^{-1}$	0.00	0.00		
36	7	6810	64	$7.33 \cdot 10^{-2}$	0.00	0.00		

Table 5.3: Measured IBER at each radio position at the UNB ITC building.

Test	Channel	Data Rate (kbps)	PRF (MHz)	R1	R2	R3	R4	R5
1	1	110	16	$3.09 \cdot 10^{-6}$	0.00	0.00	$1.25 \cdot 10^{-5}$	
2	1	110	64	0.00	0.00	0.00	$2.52 \cdot 10^{-6}$	
3	1	850	16	$1.58 \cdot 10^{-5}$	0.00	0.00	$9.55 \cdot 10^{-5}$	
4	1	850	64	$3.56 \cdot 10^{-6}$	0.00	0.00	$5.41 \cdot 10^{-5}$	
5	1	6810	16	$5.91 \cdot 10^{-3}$	$5.22 \cdot 10^{-7}$	0.00	$6.00 \cdot 10^{-2}$	
6	1	6810	64	$2.40 \cdot 10^{-3}$	0.00	0.00	$6.52 \cdot 10^{-2}$	
7	2	110	16	$1.05 \cdot 10^{-5}$	0.00	0.00	$5.98 \cdot 10^{-7}$	
8	2	110	64	$1.25 \cdot 10^{-6}$	0.00	0.00	$7.03 \cdot 10^{-7}$	
9	2	850	16	$5.24 \cdot 10^{-5}$	0.00	0.00	$1.93 \cdot 10^{-4}$	
10	2	850	64	$7.14 \cdot 10^{-5}$	0.00	0.00	$1.50 \cdot 10^{-4}$	
11	2	6810	16	$5.10 \cdot 10^{-1}$	$8.05 \cdot 10^{-7}$	$1.84 \cdot 10^{-6}$	$5.46 \cdot 10^{-2}$	
12	2	6810	64	$5.00 \cdot 10^{-1}$	0.00	$5.00 \cdot 10^{-7}$	$4.87 \cdot 10^{-2}$	
13	3	110	16	$3.29 \cdot 10^{-6}$	0.00	0.00	0.00	
14	3	110	64	$3.04 \cdot 10^{-6}$	0.00	0.00	$2.24 \cdot 10^{-7}$	
15	3	850	16	$3.02 \cdot 10^{-4}$	0.00	0.00		
16	3	850	64	$1.97 \cdot 10^{-5}$	0.00	0.00	$4.25 \cdot 10^{-4}$	
17	3	6810	16	$5.07 \cdot 10^{-1}$	0.00	0.00		
18	3	6810	64	$1.31 \cdot 10^{-1}$	0.00	0.00	$2.36 \cdot 10^{-2}$	
19	4	110	16	$2.93 \cdot 10^{-6}$	0.00	0.00	$2.44 \cdot 10^{-6}$	$1.64 \cdot 10^{-2}$
20	4	110	64	$2.17 \cdot 10^{-6}$	0.00	0.00	$2.35 \cdot 10^{-6}$	
21	4	850	16	$1.25 \cdot 10^{-5}$	0.00	0.00	$4.40 \cdot 10^{-5}$	
22	4	850	64	$6.21 \cdot 10^{-6}$	0.00	0.00	$2.50 \cdot 10^{-5}$	
23	4	6810	16	$1.47 \cdot 10^{-3}$	$4.57 \cdot 10^{-7}$	$7.62 \cdot 10^{-7}$	$5.08 \cdot 10^{-2}$	
24	4	6810	64	$3.61 \cdot 10^{-4}$	0.00	$4.57 \cdot 10^{-7}$	$6.49 \cdot 10^{-2}$	
25	5	110	16	$1.89 \cdot 10^{-6}$	0.00	0.00	0.00	
26	5	110	64	0.00	0.00	0.00	0.00	
27	5	850	16	$2.32 \cdot 10^{-5}$	0.00	0.00		
28	5	850	64	$1.26 \cdot 10^{-5}$	0.00	0.00	0.00	
29	5	6810	16	$2.25 \cdot 10^{-1}$	0.00	0.00		
30	5	6810	64	$9.63 \cdot 10^{-2}$	0.00	0.00		
31	7	110	16	$1.28 \cdot 10^{-5}$	0.00	0.00	0.00	
32	7	110	64	0.00	0.00	0.00	0.00	
33	7	850	16	$7.29 \cdot 10^{-5}$	0.00	0.00		
34	7	850	64	$1.63 \cdot 10^{-5}$	0.00	0.00	0.00	
35	7	6810	16	$4.40 \cdot 10^{-1}$	0.00	0.00		
36	7	6810	64	$1.87 \cdot 10^{-3}$	0.00	0.00		

5.2.2 UNB Heating Plant

This section presents the BER results measured at the UNB heating plant. Measurements are not available for position R4 due to a hardware failure with the mote at that position.

5.2.2.1 Packet Loss Ratio

The PLR (as defined in Section 3.1.3) for mote positions at the UNB heating plant is presented in Table 5.4.

Table 5.4: Measured PLR at each radio position at the UNB heating plant.

Test	Channel	Data Rate (kbps)	PRF (MHz)	R1	R2	R3	R5
1	1	110	16	7.66%	3.36%	98.05%	0.00%
2	1	110	64	9.74%	4.45%	88.46%	0.06%
3	1	850	16	17.06%	15.07%	99.17%	0.39%
4	1	850	64	16.68%	11.98%	94.37%	0.11%
5	1	6810	16	99.94%	99.85%	99.99%	2.27%
6	1	6810	64	99.96%	99.42%	99.98%	0.59%
7	2	110	16	53.92%	58.15%	99.30%	0.46%
8	2	110	64	64.19%	64.86%	99.62%	0.97%
9	2	850	16	81.99%	82.66%	100.00%	1.76%
10	2	850	64	89.70%	86.41%	100.00%	0.20%
11	2	6810	16	99.94%	99.91%	100.00%	29.59%
12	2	6810	64	99.96%	99.96%	100.00%	4.00%
13	3	110	16	75.48%	88.41%	98.54%	0.75%
14	3	110	64	18.41%	16.59%	58.09%	0.06%
15	3	850	16	97.92%	99.01%	100.00%	1.57%
16	3	850	64	32.42%	30.21%	87.75%	0.30%
17	3	6810	16	99.98%	99.98%	100.00%	68.97%
18	3	6810	64	99.91%	99.91%	99.94%	1.27%
19	4	110	16	8.41%	7.36%	94.90%	0.22%
20	4	110	64	9.39%	5.73%	94.54%	0.29%
21	4	850	16	20.87%	18.31%	99.79%	1.60%
22	4	850	64	15.39%	9.05%	99.63%	0.29%
23	4	6810	16	99.88%	99.33%	100.00%	13.13%
24	4	6810	64	99.86%	91.13%	99.99%	2.08%
25	5	110	16	29.00%	30.85%	94.64%	0.89%
26	5	110	64	28.56%	16.56%	80.91%	1.49%
27	5	850	16	52.11%	58.63%	98.49%	6.45%
28	5	850	64	48.58%	31.44%	91.99%	1.12%
29	5	6810	16	99.89%	99.89%	99.98%	71.80%
30	5	6810	64	99.89%	99.90%	99.96%	51.08%
31	7	110	16	69.59%	35.17%	93.25%	6.15%
32	7	110	64	34.23%	16.48%	75.51%	3.69%
33	7	850	16	92.21%	74.36%	99.84%	7.01%
34	7	850	64	42.54%	32.10%	97.73%	1.82%
35	7	6810	16	99.94%	99.93%	100.00%	65.88%
36	7	6810	64	99.88%	99.87%	99.96%	12.22%

Position R5 received the most packets, with less than 8% of packets lost, except for some test conditions using the 6.81 Mbps data rate. Position R3 encountered the most packet loss, with more than half of the motes failing to receive more than 98% of packets sent, and all test configurations losing at least 58% of packets. Positions R2 and R3 performed similarly for most test configurations.

5.2.2.2 Packet Error Ratio

Table 5.5 presents the PER (as defined in Equation 3.4) calculated for each mote position.

Table 5.5: Measured PER at each radio position at the UNB heating plant.

Test	Channel	Data Rate (kbps)	PRF (MHz)	R1	R2	R3	R5
1	1	110	16	$2.39 \cdot 10^{-5}$	$4.57 \cdot 10^{-5}$	0.00	0.00
2	1	110	64	$2.45 \cdot 10^{-5}$	0.00	$1.91 \cdot 10^{-4}$	0.00
3	1	850	16	$5.33 \cdot 10^{-4}$	$4.16 \cdot 10^{-4}$	$5.32 \cdot 10^{-3}$	0.00
4	1	850	64	$4.24 \cdot 10^{-4}$	$2.76 \cdot 10^{-4}$	$5.89 \cdot 10^{-3}$	$2.21 \cdot 10^{-5}$
5	1	6810	16	$6.43 \cdot 10^{-1}$	$4.55 \cdot 10^{-1}$	$3.33 \cdot 10^{-1}$	$3.16 \cdot 10^{-4}$
6	1	6810	64	$6.84 \cdot 10^{-1}$	$2.80 \cdot 10^{-1}$	$5.56 \cdot 10^{-1}$	$8.89 \cdot 10^{-5}$
7	2	110	16	$9.59 \cdot 10^{-5}$	$2.11 \cdot 10^{-4}$	0.00	0.00
8	2	110	64	0.00	0.00	0.00	0.00
9	2	850	16	$3.92 \cdot 10^{-3}$	$3.95 \cdot 10^{-3}$	0.00	$4.50 \cdot 10^{-5}$
10	2	850	64	$3.22 \cdot 10^{-3}$	$2.60 \cdot 10^{-3}$	0.00	0.00
11	2	6810	16	$5.93 \cdot 10^{-1}$	$6.67 \cdot 10^{-1}$		$6.87 \cdot 10^{-3}$
12	2	6810	64	$4.50 \cdot 10^{-1}$	$6.47 \cdot 10^{-1}$		$7.13 \cdot 10^{-4}$
13	3	110	16	$2.70 \cdot 10^{-4}$	$1.91 \cdot 10^{-4}$	0.00	0.00
14	3	110	64	0.00	$2.65 \cdot 10^{-5}$	$1.05 \cdot 10^{-4}$	0.00
15	3	850	16	$1.27 \cdot 10^{-2}$	$8.93 \cdot 10^{-3}$	0.00	0.00
16	3	850	64	$7.84 \cdot 10^{-4}$	$8.54 \cdot 10^{-4}$	$3.42 \cdot 10^{-3}$	$2.22 \cdot 10^{-5}$
17	3	6810	16	$5.56 \cdot 10^{-1}$	$3.64 \cdot 10^{-1}$		$2.37 \cdot 10^{-2}$
18	3	6810	64	$7.21 \cdot 10^{-1}$	$5.50 \cdot 10^{-1}$	$5.60 \cdot 10^{-1}$	$2.91 \cdot 10^{-4}$
19	4	110	16	$2.41 \cdot 10^{-5}$	$7.15 \cdot 10^{-5}$	0.00	0.00
20	4	110	64	0.00	$2.34 \cdot 10^{-5}$	0.00	0.00
21	4	850	16	$5.86 \cdot 10^{-4}$	$4.60 \cdot 10^{-4}$	$5.32 \cdot 10^{-2}$	$4.49 \cdot 10^{-5}$
22	4	850	64	$2.87 \cdot 10^{-4}$	$2.67 \cdot 10^{-4}$	$5.99 \cdot 10^{-3}$	0.00
23	4	6810	16	$6.25 \cdot 10^{-1}$	$2.83 \cdot 10^{-1}$	0.00	$2.49 \cdot 10^{-3}$
24	4	6810	64	$5.38 \cdot 10^{-1}$	$1.06 \cdot 10^{-1}$	$6.00 \cdot 10^{-1}$	$2.71 \cdot 10^{-4}$
25	5	110	16	$3.11 \cdot 10^{-5}$	$9.58 \cdot 10^{-5}$	0.00	0.00
26	5	110	64	0.00	$2.65 \cdot 10^{-5}$	$2.31 \cdot 10^{-4}$	0.00
27	5	850	16	$1.48 \cdot 10^{-3}$	$1.33 \cdot 10^{-3}$	$5.86 \cdot 10^{-3}$	$2.36 \cdot 10^{-5}$
28	5	850	64	$1.12 \cdot 10^{-3}$	$7.09 \cdot 10^{-4}$	$2.48 \cdot 10^{-3}$	$2.23 \cdot 10^{-5}$
29	5	6810	16	$6.94 \cdot 10^{-1}$	$7.31 \cdot 10^{-1}$	$7.27 \cdot 10^{-1}$	$2.15 \cdot 10^{-2}$
30	5	6810	64	$4.69 \cdot 10^{-1}$	$5.68 \cdot 10^{-1}$	$8.24 \cdot 10^{-1}$	$1.42 \cdot 10^{-2}$
31	7	110	16	$2.18 \cdot 10^{-4}$	$1.36 \cdot 10^{-4}$	$9.82 \cdot 10^{-4}$	$7.06 \cdot 10^{-5}$
32	7	110	64	$3.36 \cdot 10^{-5}$	0.00	$1.80 \cdot 10^{-4}$	0.00
33	7	850	16	$5.11 \cdot 10^{-3}$	$3.62 \cdot 10^{-3}$	$4.23 \cdot 10^{-2}$	$1.43 \cdot 10^{-4}$
34	7	850	64	$1.38 \cdot 10^{-3}$	$6.51 \cdot 10^{-4}$	$9.73 \cdot 10^{-3}$	$2.25 \cdot 10^{-5}$
35	7	6810	16	$5.19 \cdot 10^{-1}$	$5.16 \cdot 10^{-1}$	1.00	$1.94 \cdot 10^{-2}$
36	7	6810	64	$7.04 \cdot 10^{-1}$	$6.55 \cdot 10^{-1}$	$6.67 \cdot 10^{-1}$	$2.14 \cdot 10^{-3}$

5.2.2.3 Bit Error Ratio

Table 5.6 presents the IBER (as defined in Section 3.1.1) measured at each mote position.

Table 5.6: Measured IBER at each radio position at the UNB heating plant.

Test	Channel	Data Rate (kbps)	PRF (MHz)	R1	R2	R3	R5
1	1	110	16	$2.28 \cdot 10^{-6}$	$4.96 \cdot 10^{-6}$	0.00	0.00
2	1	110	64	$2.06 \cdot 10^{-6}$	0.00	$1.56 \cdot 10^{-5}$	0.00
3	1	850	16	$4.52 \cdot 10^{-5}$	$4.80 \cdot 10^{-5}$	$4.50 \cdot 10^{-4}$	0.00
4	1	850	64	$4.18 \cdot 10^{-5}$	$2.50 \cdot 10^{-5}$	$4.46 \cdot 10^{-4}$	$3.48 \cdot 10^{-6}$
5	1	6810	16	$3.45 \cdot 10^{-1}$	$4.10 \cdot 10^{-2}$	$5.10 \cdot 10^{-1}$	$7.07 \cdot 10^{-6}$
6	1	6810	64	$2.09 \cdot 10^{-1}$	$1.13 \cdot 10^{-2}$	$5.15 \cdot 10^{-1}$	$1.71 \cdot 10^{-6}$
7	2	110	16	$6.40 \cdot 10^{-6}$	$1.74 \cdot 10^{-5}$	0.00	0.00
8	2	110	64	0.00	0.00	0.00	0.00
9	2	850	16	$4.52 \cdot 10^{-4}$	$3.94 \cdot 10^{-4}$	0.00	$4.93 \cdot 10^{-6}$
10	2	850	64	$2.61 \cdot 10^{-4}$	$2.12 \cdot 10^{-4}$	0.00	0.00
11	2	6810	16	$4.98 \cdot 10^{-1}$	$4.94 \cdot 10^{-1}$		$1.58 \cdot 10^{-4}$
12	2	6810	64	$5.03 \cdot 10^{-1}$	$5.03 \cdot 10^{-1}$		$1.77 \cdot 10^{-5}$
13	3	110	16	$3.36 \cdot 10^{-5}$	$3.10 \cdot 10^{-5}$	0.00	0.00
14	3	110	64	0.00	$3.78 \cdot 10^{-6}$	$3.11 \cdot 10^{-6}$	0.00
15	3	850	16	$1.35 \cdot 10^{-3}$	$7.36 \cdot 10^{-4}$	0.00	0.00
16	3	850	64	$8.44 \cdot 10^{-5}$	$6.71 \cdot 10^{-5}$	$3.08 \cdot 10^{-4}$	$2.29 \cdot 10^{-6}$
17	3	6810	16	$4.75 \cdot 10^{-1}$	$4.86 \cdot 10^{-1}$		$5.31 \cdot 10^{-4}$
18	3	6810	64	$4.96 \cdot 10^{-1}$	$4.93 \cdot 10^{-1}$	$5.06 \cdot 10^{-1}$	$6.23 \cdot 10^{-6}$
19	4	110	16	$1.80 \cdot 10^{-6}$	$4.88 \cdot 10^{-6}$	0.00	0.00
20	4	110	64	0.00	$3.93 \cdot 10^{-7}$	0.00	0.00
21	4	850	16	$5.95 \cdot 10^{-5}$	$4.47 \cdot 10^{-5}$	$6.30 \cdot 10^{-3}$	$2.67 \cdot 10^{-6}$
22	4	850	64	$3.46 \cdot 10^{-5}$	$3.27 \cdot 10^{-5}$	$1.93 \cdot 10^{-4}$	0.00
23	4	6810	16	$3.66 \cdot 10^{-1}$	$1.20 \cdot 10^{-2}$		$5.31 \cdot 10^{-5}$
24	4	6810	64	$7.66 \cdot 10^{-2}$	$2.75 \cdot 10^{-3}$	$5.46 \cdot 10^{-1}$	$5.24 \cdot 10^{-6}$
25	5	110	16	$3.11 \cdot 10^{-6}$	$6.69 \cdot 10^{-6}$	0.00	0.00
26	5	110	64	0.00	$1.66 \cdot 10^{-6}$	$1.76 \cdot 10^{-5}$	0.00
27	5	850	16	$1.42 \cdot 10^{-4}$	$1.30 \cdot 10^{-4}$	$3.98 \cdot 10^{-4}$	$3.97 \cdot 10^{-6}$
28	5	850	64	$1.01 \cdot 10^{-4}$	$7.67 \cdot 10^{-5}$	$2.34 \cdot 10^{-4}$	$2.42 \cdot 10^{-6}$
29	5	6810	16	$5.01 \cdot 10^{-1}$	$4.85 \cdot 10^{-1}$	$5.10 \cdot 10^{-1}$	$5.28 \cdot 10^{-4}$
30	5	6810	64	$4.97 \cdot 10^{-1}$	$4.95 \cdot 10^{-1}$	$5.00 \cdot 10^{-1}$	$3.15 \cdot 10^{-4}$
31	7	110	16	$1.89 \cdot 10^{-5}$	$1.72 \cdot 10^{-5}$	$6.41 \cdot 10^{-5}$	$4.68 \cdot 10^{-6}$
32	7	110	64	$5.22 \cdot 10^{-6}$	0.00	$1.50 \cdot 10^{-5}$	0.00
33	7	850	16	$4.72 \cdot 10^{-4}$	$3.11 \cdot 10^{-4}$	$1.01 \cdot 10^{-3}$	$1.31 \cdot 10^{-5}$
34	7	850	64	$1.21 \cdot 10^{-4}$	$4.56 \cdot 10^{-5}$	$1.09 \cdot 10^{-3}$	$2.63 \cdot 10^{-6}$
35	7	6810	16	$5.16 \cdot 10^{-1}$	$5.13 \cdot 10^{-1}$	$3.50 \cdot 10^{-1}$	$4.86 \cdot 10^{-4}$
36	7	6810	64	$5.08 \cdot 10^{-1}$	$4.56 \cdot 10^{-1}$	$5.06 \cdot 10^{-1}$	$4.34 \cdot 10^{-5}$

5.3 Mesh Network with OpenWSN and WirelessHART

Two experiments were performed on both the OpenWSN and WirelessHART networks. The first experiment measured the upstream latency from the motes to the base station. The second experiment measures the RTT from the base station to the motes. They are performed in parallel on both the OpenWSN and WirelessHART networks, in order to subject both tests to the same RF conditions which may vary over time. The networks do not interfere with each other since the OpenWSN and WirelessHART networks do not operate on overlapping frequency bands (see Sections 2.1.1 and 3.2).

For the upstream experiments, the measurements for all 5 motes are performed in parallel, i.e. all five motes on both networks are publishing a packet every 2 seconds. This simulates a real-world setup where multiple motes would transfer data on the network at the same time. The motes do not necessarily generate packets at exactly the same time since the application-level timers used to generate the packets on the mote are not synchronized across motes.

The upstream tests are performed over 900 packets that are sent at a rate of 1 packet every 2 seconds, resulting in an overall test duration of 30 minutes. During initial testing, sending 1800 packets at a rate of 1 packet every 1 second was attempted. The one packet per second rate test could not be completed successfully due to a failure in the OpenWSN network. Mote #3 failed after sending approximately 1700 packets Consequently, data for motes #3, #4, and #5 could not be retrieved for the remainder of the test duration. The retrieved samples leading up to the failure showed extreme latency of several minutes, where older packets had been queued. It is suspected that the failure was caused by the packet queue at mote #3 becoming full, which caused the mote to be unable to allocate memory buffers

to send important packets, such as beacons. OpenWSN is intended as an open-source implementation of protocols for the Internet of Things. Its implementation of the protocol stack is primarily a research platform for testing viability of the used protocols in many settings, including industrial control. OpenWSN could be improved to increase reliability in the presence of faults, errors, and traffic exceeding the availble network capacity. Appendix J provides a detailed descripton of reliability issues that occurred during the testing in this thesis.

For the RTT measurements, 600 packets are sent at a rate of 1 packet every 5 seconds, resulting in a test duration of 50 minutes. A slower update rate than 1 second was required due to queuing at the WirelessHART network manager.

5.3.1 Packet Acknowledgement Ratio

The PAR is measured after all tests have completed. The WirelessHART PAR is measured by the network manager and is reported for each active upstream and downstream link in the network. The network manager only reports the PAR for mote-to-mote links and manager-to-mote downstream links; it does not report the PAR for upstream mote-to-manager links.

Each OpenWSN mote measures the PAR in two ways. Firstly, the mote counts N_{tx} and N_{txAck} for packets sent to each neighboring mote using any available slot, i.e. packets sent to the neighbor over both shared and non-shared links are counted. Secondly, the mote separately counts N_{tx} and N_{txAck} for each packet sent in a specific slot offset and channel offset. This permits the link PAR to be separately calculated for each shared and non-shared link. The PAR is calculated from the N_{tx} and N_{txAck} values as shown in Equation 4.4. The network schedules used by OpenWSN during the experiments are provided in Appendix K.

5.3.1.1 UNB ITC Building

Figure 5.3 shows the PAR measured for each link used in the WirelessHART network. The mote positions in Figure 5.3 are laid out according to their approximate real-world positions shown in Figure 4.1.

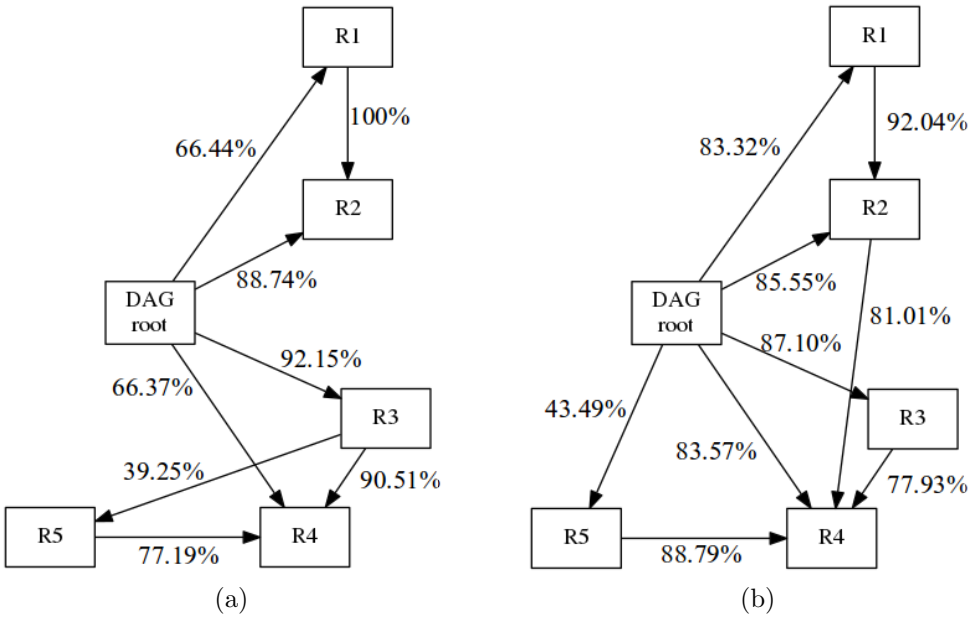


Figure 5.3: WirelessHART PAR at the UNB ITC building after (a) upstream tests (b) round trip tests.

As shown in Figure 5.3 the network manager was able to communicate directly with all motes in the network during the round trip tests. For the upstream tests the network manager communicated directly with all motes except R5.

Figures 5.4 and 5.5 show the PAR calculated for all packets sent to each neighboring mote in the upstream and downstream directions respectively. The neighbor PAR is calculated over packets sent to the neighbor using any link, i.e. using a GTS or shared slot. The PAR is not available for all downstream links as N_{tx} and N_{txAck} are not counted by OpenWSN for all downstream links.

Comparing the upstream and downstream PAR between motes R2 and R3 to the DAG root show that upstream communication is much more reliable than down-

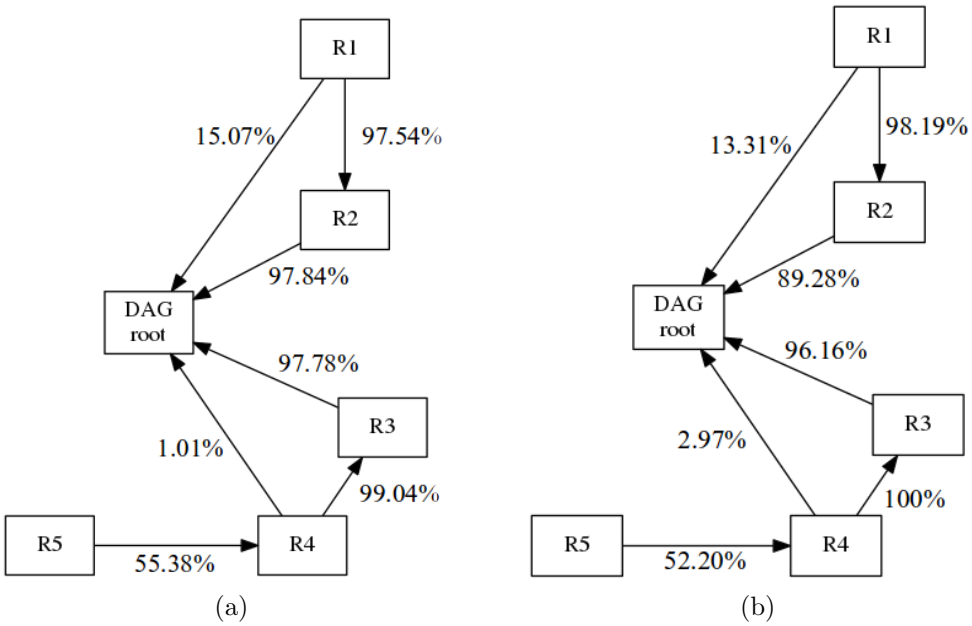


Figure 5.4: OpenWSN upstream neighbor PAR at the UNB ITC building after (a) upstream tests (b) round trip tests.

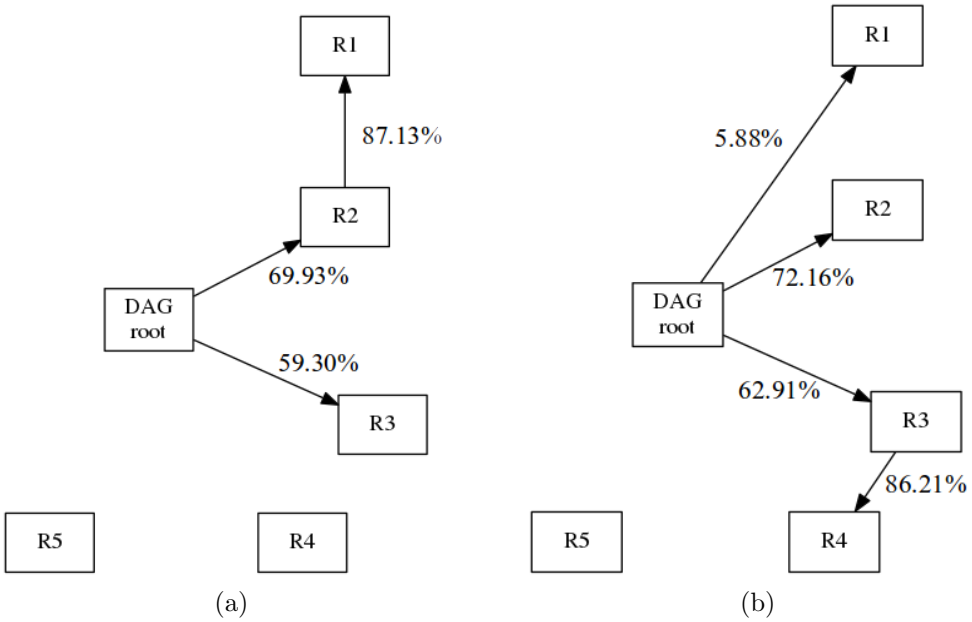


Figure 5.5: OpenWSN downstream neighbor PAR at the UNB ITC building after (a) upstream tests (b) round trip tests.

stream communication between the same motes. For example, for upstream communication from mote R3 to the DAG root approximately 98% of packets were suc-

cessfully acknowledged. However, in the downstream direction only 63% of packets were successfully acknowledged.

Figure 5.6 shows the PAR for each GTS in the network schedule. Some motes have multiple GTS to the same mote, in which case the PAR for each individual GTS is shown.

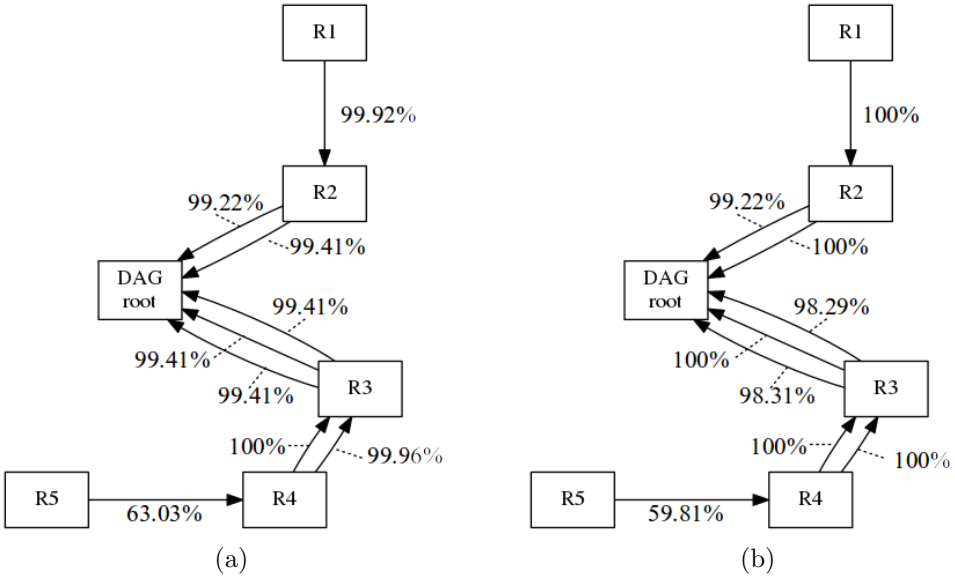


Figure 5.6: OpenWSN schedule link PAR at the UNB ITC building after (a) upstream tests (b) round trip tests.

The upstream PAR for all OpenWSN is >99% for all motes except for the upstream link from R5 to R4, where the PAR was measured between 60% and 63% for the two test scenarios. A comparison between the upstream and downstream links between the DAG root and motes R3 and R4 reveals a significant difference in the reliability of the link due to the use of GTS in the upstream direction only. In the downstream direction there is contention between the DAG root and the other motes attempting to use the shared slot at the same time, causing a reduction in the PAR in the downstream direction.

Motes may also use the shared slot to transmit packets upstream, which increases the contention in the shared slot and reduces the PAR to the neighbor. For example,

in the upstream link from mote R3 to the DAG root the PAR for packets sent using both GTS and the shared slot shown in Figure 5.4 is 97% ±1%. However, when only using GTS as shown in Figure 5.6 the PAR is 99% ±1%.

The schedule links shown in Figure 5.6 indicate that motes R2 and R3 are 1 hop neighbors to the DAG root, and motes R1 and R4 are 2 hops away. Mote R5 is 3 hops from the DAG root, as messages are routed through R4 and R3.

5.3.1.2 UNB Heating Plant

The WirelessHART network manager was able to form a network where all motes were 1 hop, i.e. no routing was necessary. The network manager did, however, allocate additional links to improve the reliability of the network. Figure 5.7 shows the network topology that was formed by the network manager, and the calculated PAR for each link.

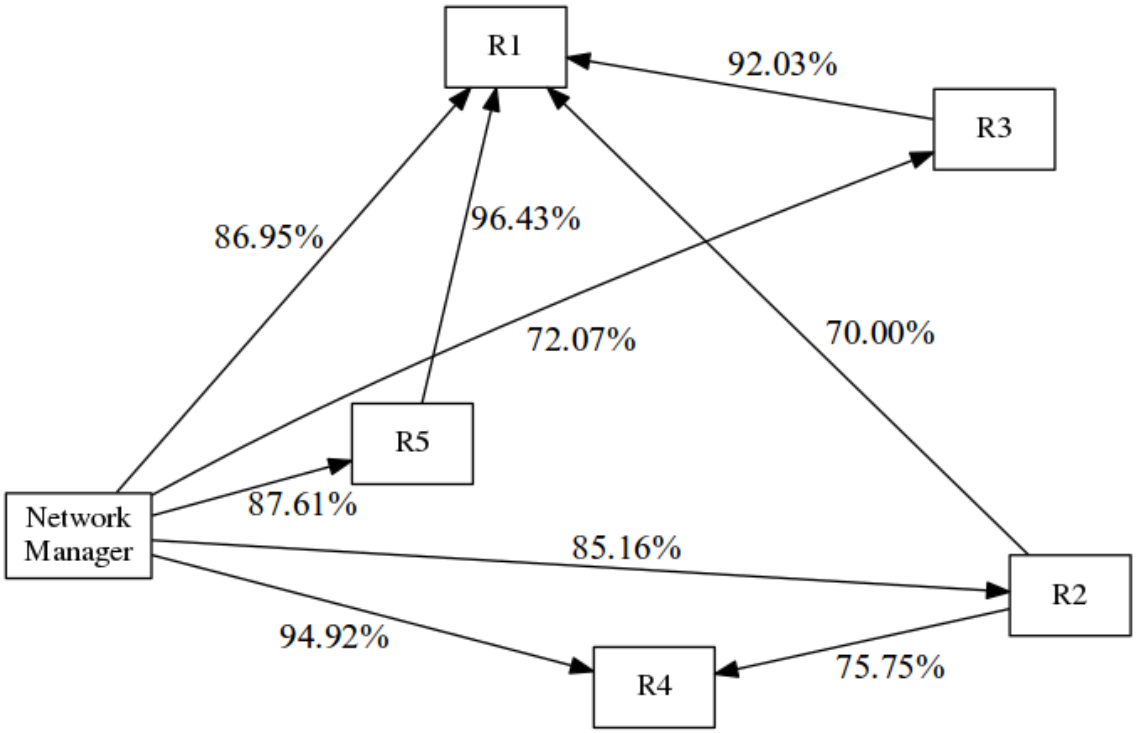


Figure 5.7: WirelessHART link PAR at the UNB heating plant after the upstream and round-trip test scenarios.

The WirelessHART network achieved good reliability for all links in the network, with the minimum PAR of 70% for the upstream communication from R2 to R1. Direct communication to R3 was also reliable with a PAR of 72%, even though there were two large boilers obstructing the straight-line path between the network manager and the mote.

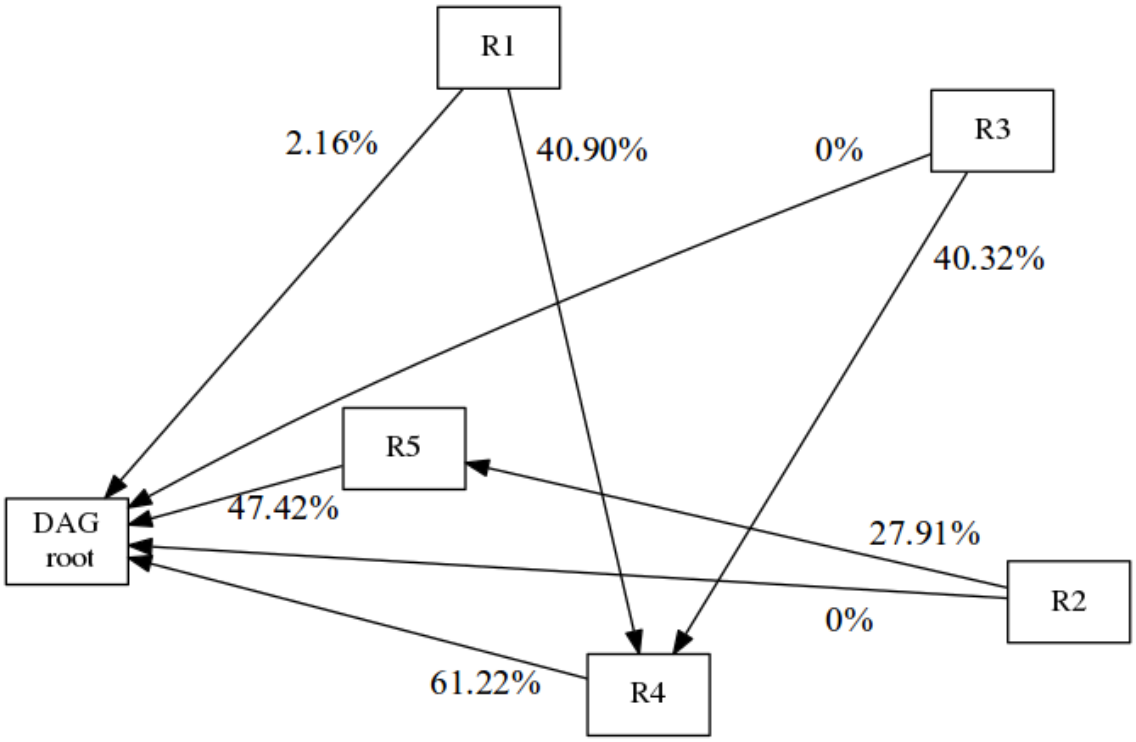


Figure 5.8: OpenWSN upstream neighbor PAR at the UNB heating plant after both test scenarios.

The OpenWSN PAR calculated for all packets sent to each neighboring mote is shown in Figures 5.8 and 5.9 for the upstream and downstream directions respectively. The UWB network is not able to achieve the same communication range as WirelessHART, and depends on routing messages through motes R4 and R5. Some communication was attempted from mote R3 to the DAG root, but no packets were successfully received. Similarly, communication from R1 to the DAG root was also attempted. However, only 2% of the packets sent from R1 were successfully acknowledged.

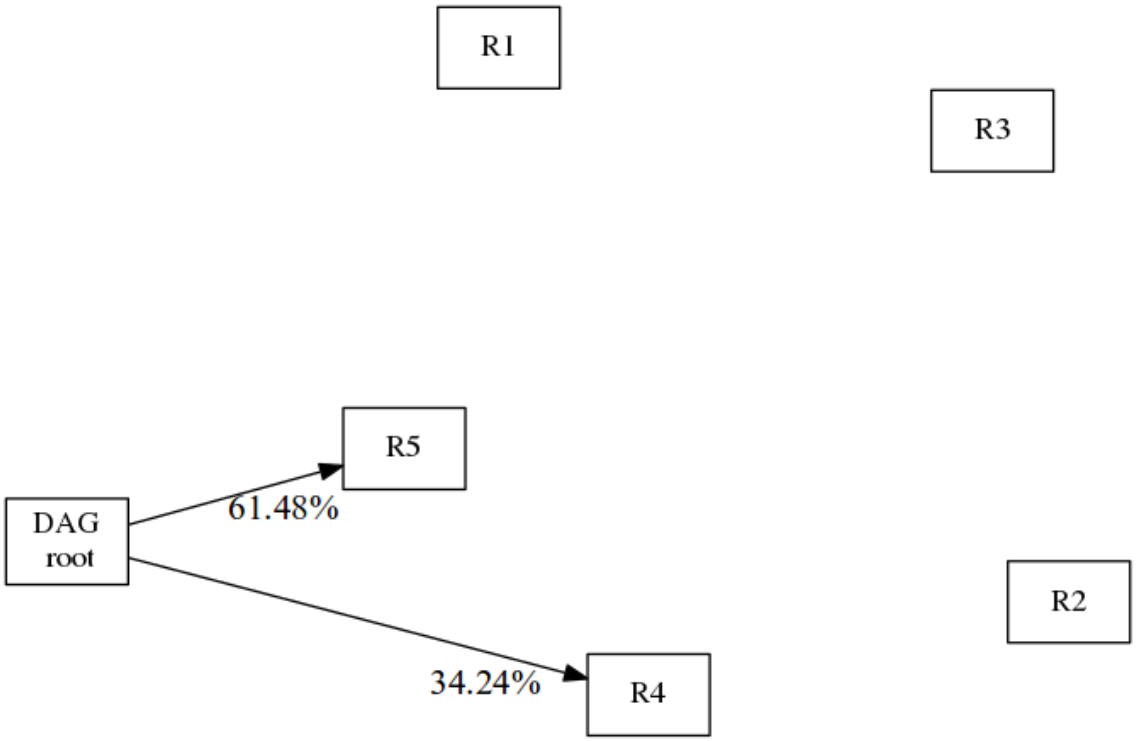


Figure 5.9: OpenWSN downstream neighbor PAR at the UNB heating plant after both test scenarios.

OpenWSN allocated two dedicated links in the schedule for mote R5 to transmit to the DAG root, as shown in Figure 5.10. Due to a fault in OpenWSN, there was a mismatch between the DAG root and mote R5 schedules where mote R5 had both links in its schedule and the DAG root only had one of the links in its schedule. Therefore, mote R5 attempted to use both links for sending packets but the DAG root only enabled its receiver during one link. This resulted in 100% packet loss for one of the links as shown in Figure 5.10. The second link was correct in both schedules, and achieved a 70.12% PAR.

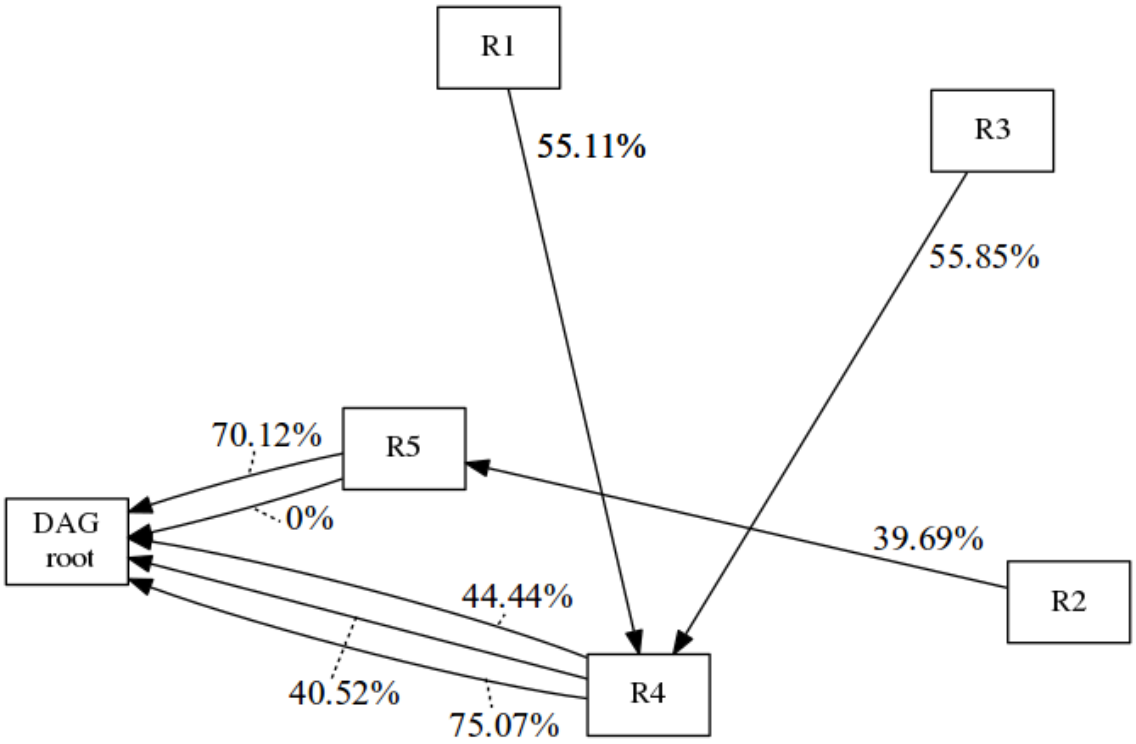


Figure 5.10: OpenWSN non-shared link PAR at the UNB heating plant after the upstream and round-trip test scenarios.

5.3.2 Latency

OpenWSN does not appear to detect or filter out duplicate packets that are received, and a number of duplicate packets were received during the tests. The latency measurements shown in this section do not include duplicated packets. The WirelessHART network filtered out any duplicate packets.

The latency measurements presented in this section are represented using box plots [35]. Plots for the latency of individual samples are presented in Appendix L. The average latency measurements are included in Appendix M.

5.3.2.1 UNB ITC Building

Figure 5.11 and Figure 5.12 illustrates the upstream latency measurements for the upstream test scenario. The latency measurements reflects the number of hops to the gateway, with the median latency increasing proportionally to the number of hops.

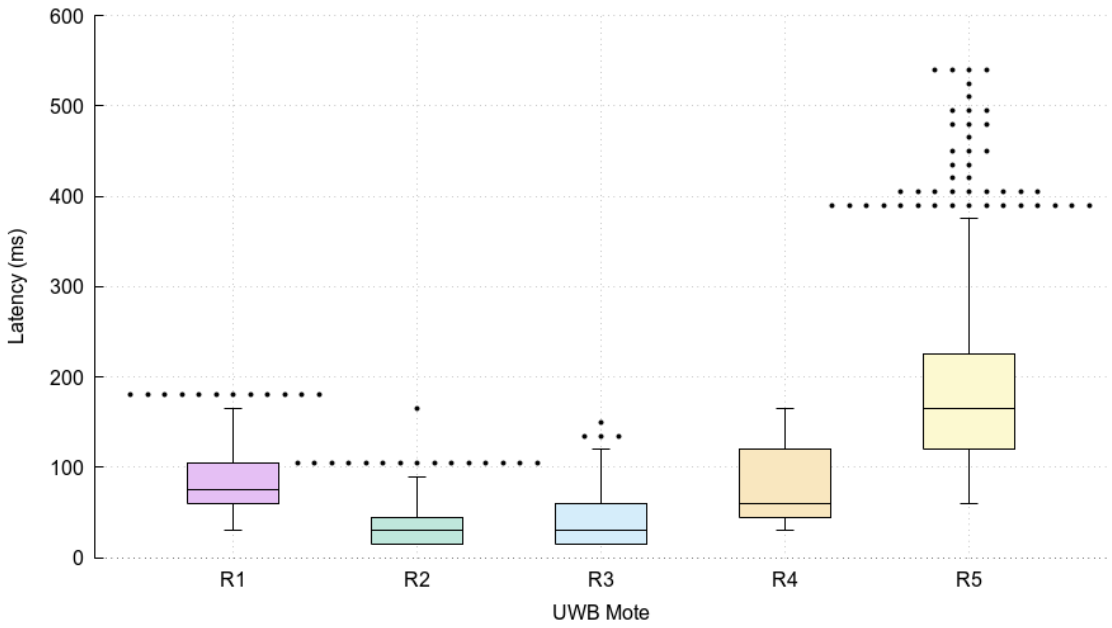


Figure 5.11: Box plots for the UWB motes' upstream latency at the UNB ITC building.

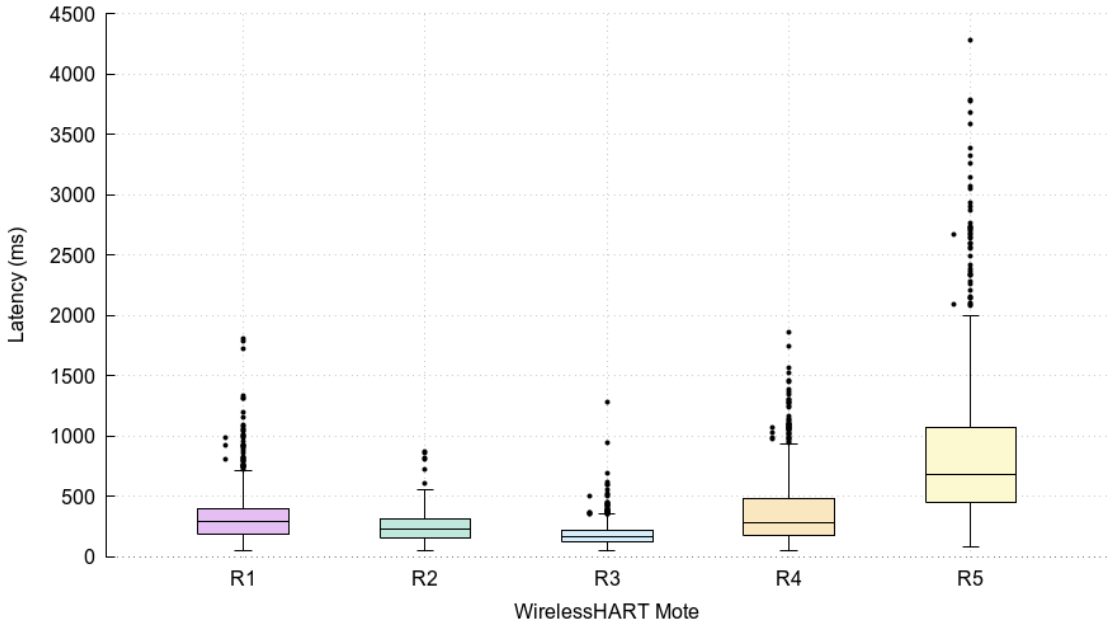


Figure 5.12: Box plots for the WirelessHART motes' upstream latency at the UNB ITC building.

For all mote positions the UWB OpenWSN network achieved lower latency than WirelessHART in both upstream and downstream directions.

The RTT measurements are shown in Figures 5.13 and 5.14 for the UWB motes and WirelessHART motes respectively.

As illustrated in Figure 5.14, the RTT recorded for the WirelessHART mote at position R5 is significantly higher than all other motes. This was caused by queuing delays at the network manager due to retransmission attempts of previous packets. During the test the queuing delay gradually increased over the course of 100 samples causing the RTT to peak at 146 seconds. The queuing delay then decreased gradually as network conditions improved. The average upstream and downstream latency did not increase during this time. Figure 5.15 shows the upstream, downstream, and round trip latency measurements for each sample during the test which illustrates this behaviour.

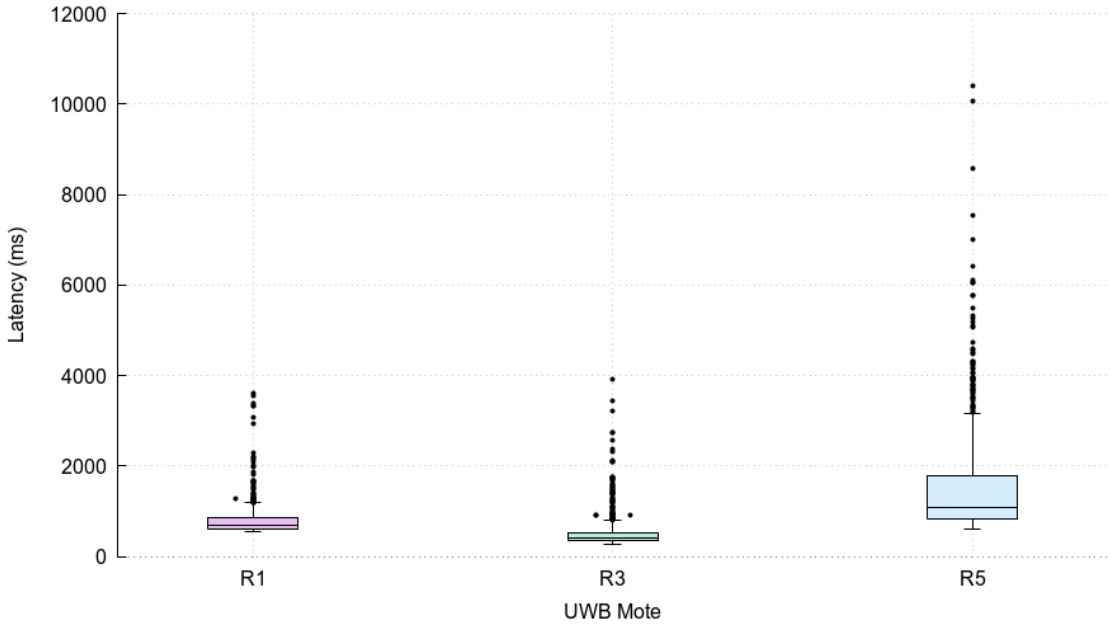


Figure 5.13: Box plots for the UWB motes' RTT at the UNB ITC building.

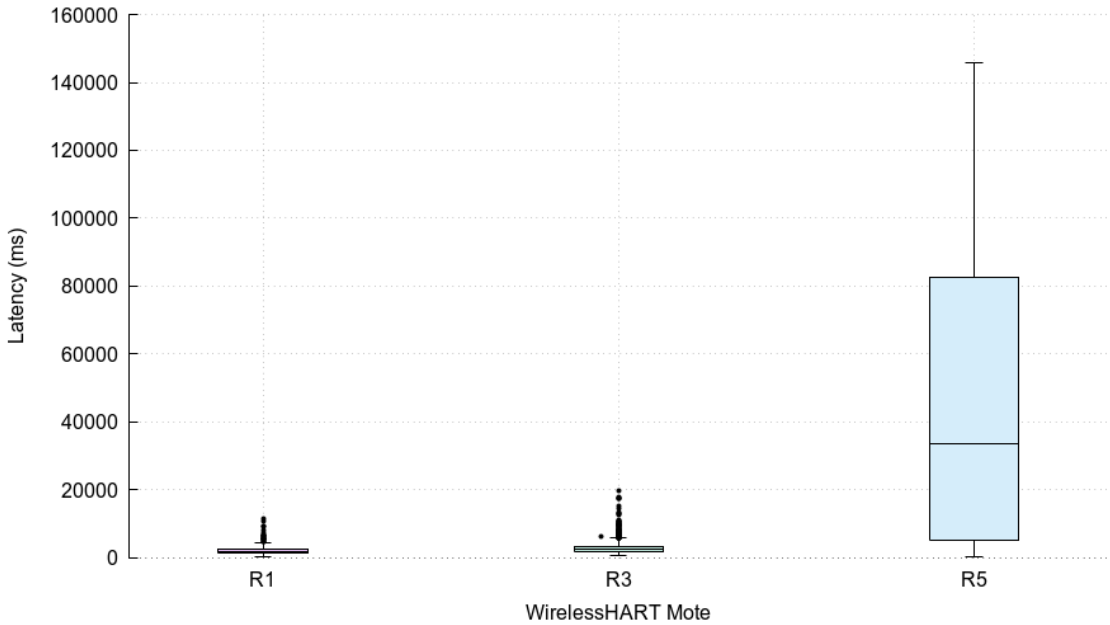


Figure 5.14: Box plots for the WirelessHART motes' RTT at the UNB ITC building.

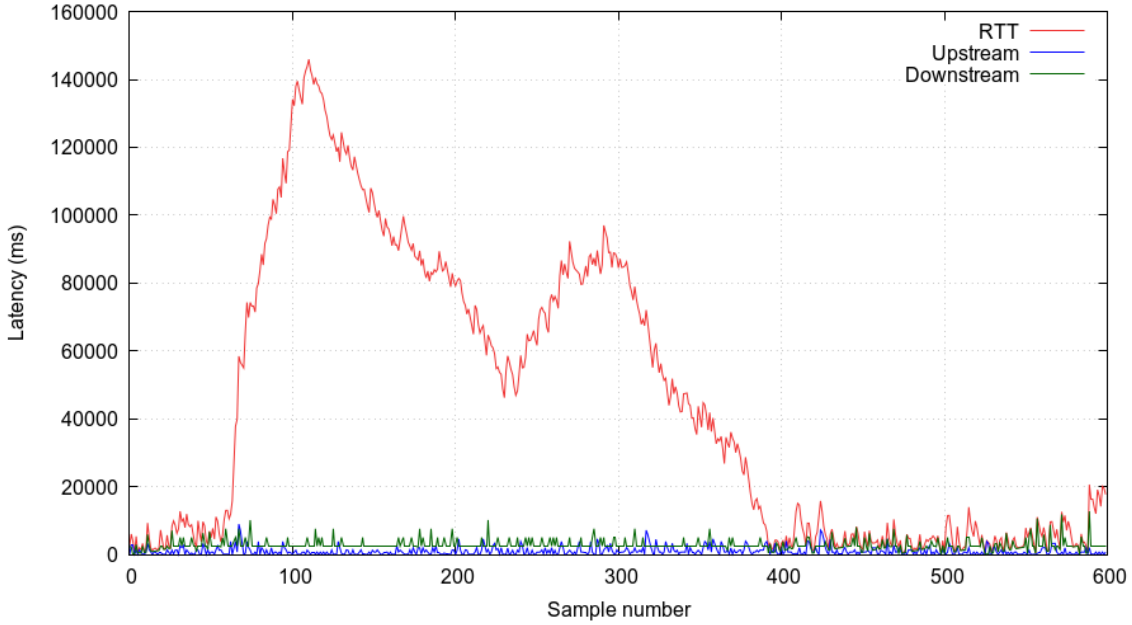


Figure 5.15: Upstream, downstream, and round trip latency measurements for WirelessHART mote R5 at the UNB ITC building.

5.3.2.2 UNB Heating Plant

The upstream test latency for both the UWB OpenWSN and WirelessHART motes is presented in Figures 5.16 and 5.17.

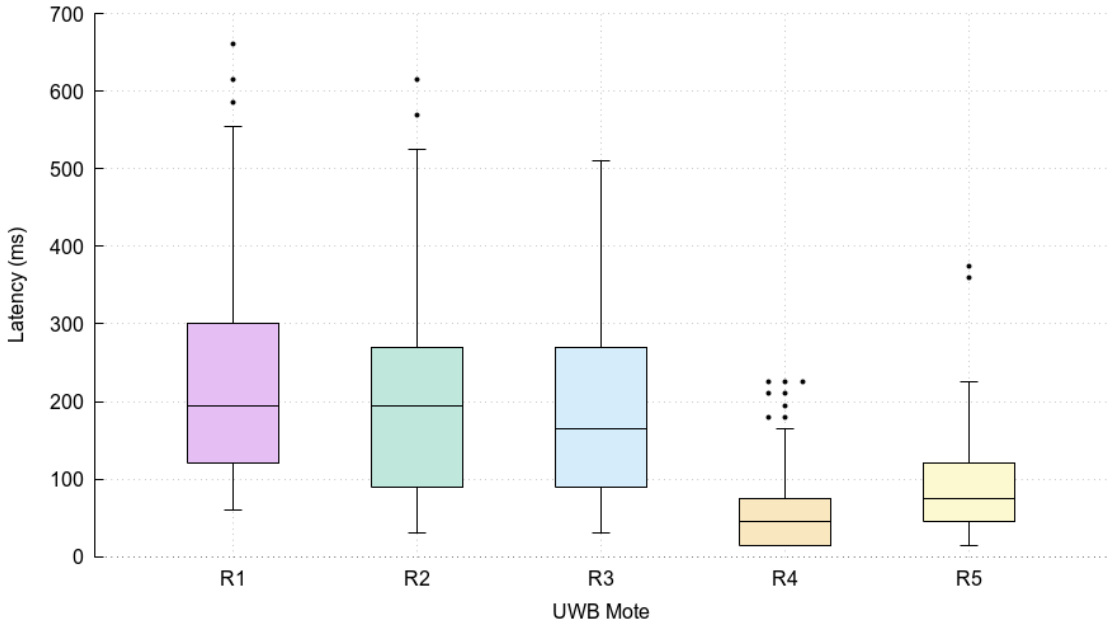


Figure 5.16: Box plots for the UWB motes' upstream latency at the UNB heating plant.

Similarly to the upstream tests performed at the UNB ITC building, the upstream latency reflects the number of hops. Although UWB motes R4 and R5 are 1 hop from the DAG root, R5 has slightly higher latency than R4 due to the fault with one of the network schedule links as described in Section 5.3.1.2, resulting in decreased communication reliability to the DAG root.

Figures 5.18 and 5.19 illustrate the RTT at the UNB heating plant for the UWB and WirelessHART motes respectively. For both motes, the UWB network achieves lower latency than the WirelessHART motes.

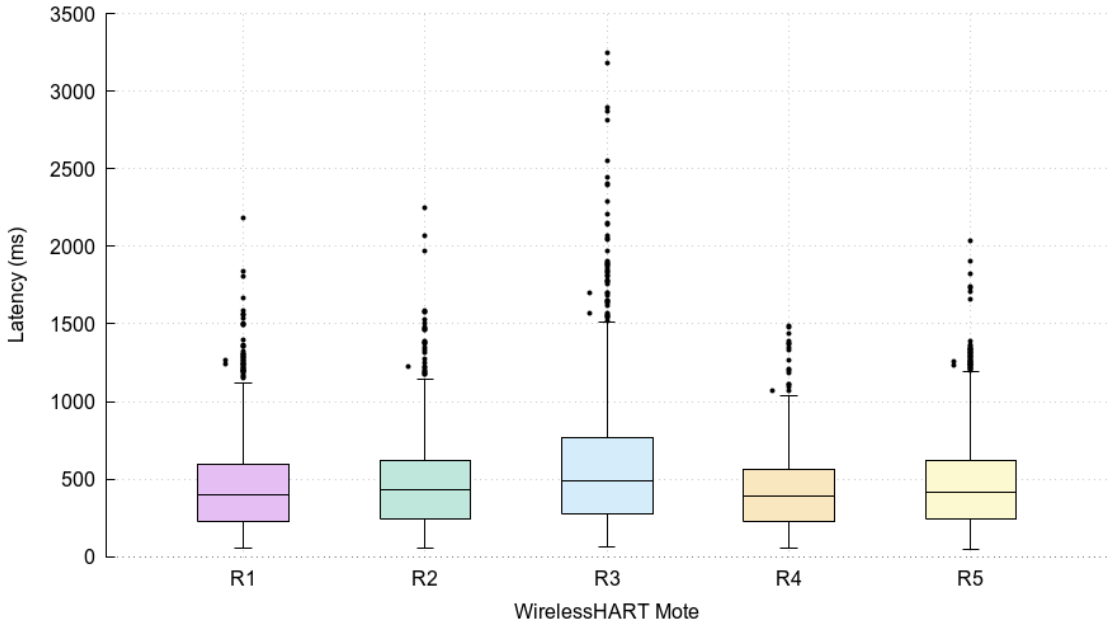


Figure 5.17: Box plots for the WirelessHART motes' upstream latency at the UNB heating plant.

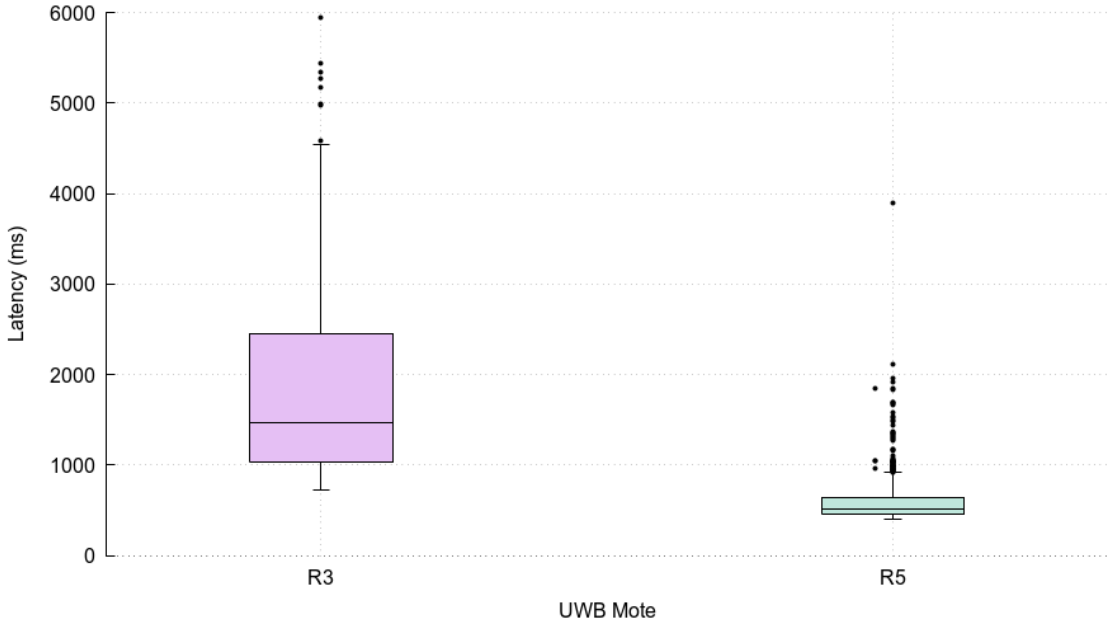


Figure 5.18: Box plots for the UWB motes' RTT at the UNB heating plant.

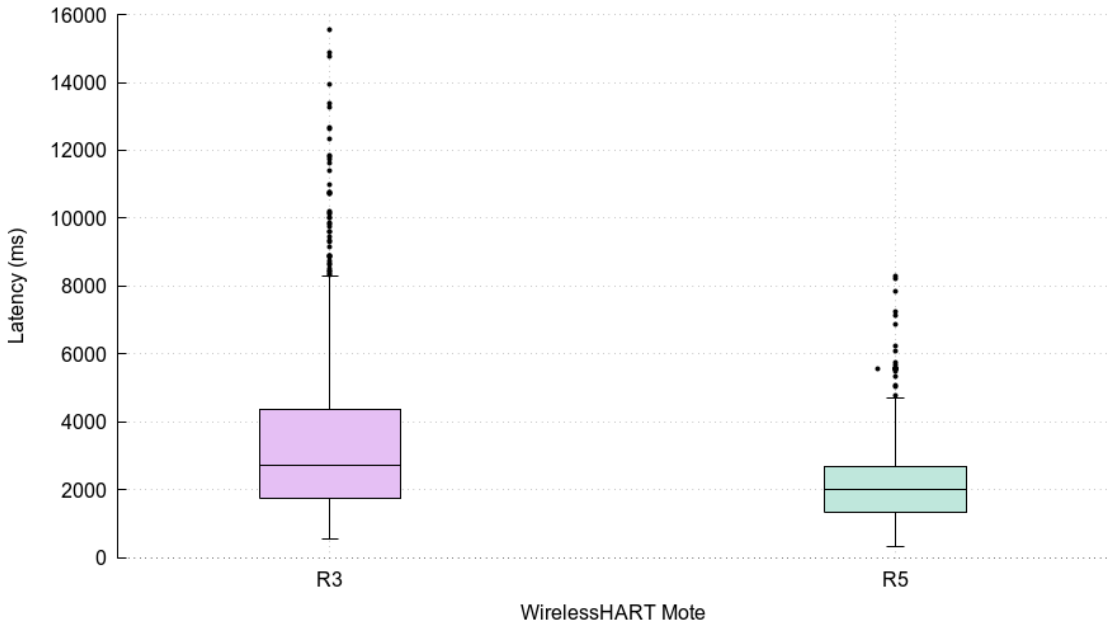


Figure 5.19: Box plots for the WirelessHART motes' RTT at the UNB heating plant.

5.3.3 Packet Loss Ratio

Some packets did not arrive at their destination during the upstream and round trip test scenarios. Table 5.7 shows the number of lost packets for the upstream test scenario at both test locations. Table 5.8 shows the number of lost packets for the round trip test scenario at both test locations.

Table 5.7: Number of upstream packets lost out of 900 packets sent.

	Mote #1		Mote #2		Mote #3		Mote #4		Mote #5	
	UWB	WH	UWB	WH	UWB	WH	UWB	WH	UWB	WH
ITC building	0	0	0	0	22	0	3	0	23	0
Heating plant	116	0	250	0	55	0	62	0	55	0

Table 5.8: Number of round trip packets lost out of 600 packets sent.

	Mote #1		Mote #3		Mote #5	
	UWB	WH	UWB	WH	UWB	WH
ITC building	3	1	11	4	17	5
Heating plant			174	1	40	0

At the UNB ITC building the WirelessHART network achieved 0% packet loss for all upstream packets. In comparison, the UWB OpenWSN network lost fewer than 2.56% of packets in the worst case for the mote at position R5. In the industrial environment at the UNB heating plant WirelessHART again achieved 0% upstream packet loss. The UWB OpenWSN network could not achieve a 0% PLR, losing between 6.11% – 27.78% of packets.

During the round trip tests only one mote achieved 0% packet loss at mote R5 at the UNB heating plant. Otherwise, up to 0.83% of packets were lost for WirelessHART and 29% were lost for UWB OpenWSN.

5.4 Discussion

5.4.1 OpenWSN for Industrial Wireless Networks

OpenWSN configures the network schedule with a single shared slot which is used for all network synchronization packets and downstream communication. The use of a single shared slot causes collisions when multiple motes attempt to use the slot simultaneously. This negatively affects the reliability of downstream communication and causes increased latency and packet loss. Upstream communication is more reliable as OpenWSN allocates GTSs for each mote in the upstream direction.

The downstream latency and reliability of OpenWSN could be improved by allocating GTS for downstream traffic to motes. The number of GTS to allocate would depend on the bandwidth requirements of the target application. Furthermore, the number of shared slots could be increased to reduce the contention and reduce latency and packet loss.

5.4.2 UWB Performance

The UWB BER measurements in Section 5.2 show that the PLR, PER, and IBER are highly variable depending on the data rate, with lower data rates generally producing fewer errors. This is reflected in Tables 5.9 and 5.10 which show the IBER computed over the combined measurements from all channels and PRFs for each data rate at the UNB ITC building and UNB heating plant, respectively. The combined IBER is calculated using Equation 3.1 where ε and n are the sum of incorrect bits and total bits received, respectively, from all test configurations using the same data rate.

These results show that the IBER can improve by at least an order of magnitude as the data rate is decreased. However, decreasing the data rate requires more time to transmit each packet. Slower data rates also use more energy as the transmitter and receiver are active for a longer duration which has a negative impact on battery-

powered devices.

Table 5.9: Combined IBER measurements at the UNB ITC building for each UWB data rate.

Data rate (kbps)	R1	R2	R3	R4	R5
110	$3.30 \cdot 10^{-6}$	0.00	0.00	$6.25 \cdot 10^{-6}$	$5.21 \cdot 10^{-1}$
850	$3.20 \cdot 10^{-5}$	0.00	0.00	$8.73 \cdot 10^{-5}$	
6810	$1.49 \cdot 10^{-3}$	$2.96 \cdot 10^{-7}$	$2.96 \cdot 10^{-7}$	$4.67 \cdot 10^{-1}$	

Table 5.10: Combined IBER measurements at the UNB heating plant for each UWB data rate.

Data rate (kbps)	R1	R2	R3	R5
110	$3.51 \cdot 10^{-6}$	$4.74 \cdot 10^{-6}$	$1.15 \cdot 10^{-5}$	$3.71 \cdot 10^{-7}$
850	$9.67 \cdot 10^{-5}$	$7.63 \cdot 10^{-5}$	$4.19 \cdot 10^{-4}$	$2.90 \cdot 10^{-6}$
6810	$2.71 \cdot 10^{-1}$	$7.50 \cdot 10^{-3}$	$5.07 \cdot 10^{-1}$	$9.87 \cdot 10^{-5}$

The results also show that the PRF can have a large impact on the frequency of transmission errors, with the 64 MHz PRF generally performing better than the 16 MHz PRF. The largest difference in PLR measured between the 16 MHz and 64 MHz PRF was measured at the UNB heating plant at position R2 on channel 3 at a data rate of 110 kbps, where the PLR was 88.41% with a 16 MHz PRF and 16.59% with a PRF of 64 MHz.

At the UNB ITC building the UWB mesh network provided very high PARs of better than 98% for dedicated links, except for the link between motes R5 and R4 where at most 63.03% PAR was achieved with UWB. In the industrial environment at the UNB heating plant, the UWB mesh network could only achieve up to 75% PAR with dedicated slots. The link reliability could be improved by using the 110 kbps data rate. This would require increasing the 15 ms time slot duration in OpenWSN to 30 ms due to the increased time required to transmit the packet, which could increase network latency.

5.4.3 WirelessHART and UWB Comparison

Experimental results show that the WirelessHART network provides higher reliability than the UWB OpenWSN network for all mote positions as more packets were lost in the OpenWSN network. The OpenWSN network, however, achieved lower average latency for all mote positions.

For GTS links between motes in the office environment, the UWB OpenWSN network achieved better link PAR for GTSs than WirelessHART. In the industrial environment the UWB network could not achieve better than 75% PAR for any GTS, whereas WirelessHART had between 70% to 95% PAR. This resulted in more reliable communication in the WirelessHART network in the industrial environment. Table 5.11 illustrates the performance difference for UWB and WirelessHART in the office and industrial environments. WirelessHART achieves worse performance in the office environment than in the industrial environment. This may be caused by closer proximity to the two WiFi access points in the office environment than in the industrial environment. The office environment also has less open space than the industrial environment, which may affect radio propagation for narrowband WirelessHART signals.

Table 5.11: Average GTS PAR for UWB and WirelessHART.

Environment	UWB	WirelessHART
Office	95.55%	71.21%
Industrial	54.4%	84.55%

The communication range of WirelessHART exceeds UWB in all environments. This is particularly visible in the industrial environment where all WirelessHART motes were reachable in a single hop from the network manager, whereas the UWB OpenWSN network required motes R4 and R5 to route messages for the other motes. Similarly, WirelessHART showed better reliability of packets sent from mote R5 to R4 than UWB OpenWSN in the office environment.

The increased range of WirelessHART is likely due to the increased transmit power that is allowed by WirelessHART compared to UWB. The UWB radios are limited to -41.3 dBm/MHz by the emissions limit imposed by the FCC and Industry Canada as described in Section 2.3.3. For channels utilizing a 500 MHz bandwidth this results in a maximum transmit power of -14.3 dBm. Using their default configuration the WirelessHART motes transmit at a power of +8 dBm with an antenna gain of +2 dBi. Furthermore, the WirelessHART network operates in a lower frequency band than UWB which may provide better penetration through line-of-sight obstacles.

Power consumption is another concern for battery-powered devices in a WSN [24]. Techniques for reducing power consumption has been largely studied in the literature. The power consumption of the radio hardware also plays a role. Table 5.12 compares the current consumption for the SmartMesh WirelessHART LTC5800-WHM mote-on-chip and the DecaWave DW1000 transceiver. The DW1000 higher current consumption than the LTC5800-WHM for both transmit and receive modes. The higher power consumption of the DW1000 may be influenced by the faster clocks required for the UWB radio, as well as increased signal processing in the UWB radio for the UWB ranging implementation.

Table 5.12: Typical power consumption of the LTC5800-WHC mote-on-chip and the DecaWave DW1000 (from [2, 16]).

	Transmit (mA)	Receive (mA)	Deep sleep (μA)
LTC5800-WHC	5.6 (+0 dBm) 9.9 (+8 dBm)	4.7	0.8
DW1000	89	123	0.2

Chapter 6

Conclusions

This thesis investigated the use of ultra-wideband (UWB) radio for use in industrial WSNs based on IEEE 802.15.4e time slotted channel hopping (TSCH). The bit error ratio (BER) characteristics of UWB in an industrial environment were experimentally measured using the DecaWave DW1000 UWB transceiver for different combinations of data rate, pulse repetition frequency (PRF), and communication frequency. We measured a maximum information bit error ratio (IBER) of $5.31 \cdot 10^{-4}$ in an industrial environment and $3.61 \cdot 10^{-4}$ in an indoor office/laboratory environment when no more than 80% of packets were lost.

We adapted the OpenWSN TSCH framework to operate using the DecaWave-based UWB physical layer. To our knowledge, this is the first time an IEEE 802.15.4e compliant system has been adapted to use UWB communication. The performance of UWB in a mesh network was measured and compared against the industry standard WirelessHART technology. Two experimental scenarios were completed to measure the latency and reliability for wireless sensor networks (WSNs), and round trip latency and reliability for wireless sensor and actuator networks (WSANs).

For the UWB-based network in an office environment, 95.55% of packets sent using guaranteed time slots (GTSs) were successfully acknowledged without errors,

resulting in less than 3% packet loss. In comparison, the WirelessHART network successfully transmitted 71.21% of packets without errors with 0% packet loss in the upstream direction and less than 1% loss for round-trip packets.

The industrial environment proved to be a more harsh environment for UWB and resulted in an average of 54.4% of packets successfully acknowledged, with an average of 12% upstream packets lost and an 18% loss for round trip packets. WirelessHART achieved a better reliability in the industrial environment with an average of 84.55% of packets successfully acknowledged, with 0% loss in the upstream direction and only a single round trip packet lost. Furthermore, the range of the WirelessHART network exceeded the UWB communication range, allowing for direct communication with all motes whereas the UWB network required routing motes in order to reach distant motes.

Compared to WirelessHART, UWB seems to be relatively fast (approximately half the upstream latency), but loses more packets (12% of all upstream packets lost) in an industrial setting. In the office environment where the distance between motes is shorter, UWB lost 1% of all upstream packets. In our opinion, UWB is suitable for wireless control. The range for reliable communication, however, may be shorter than WirelessHART.

The IBER measurements in Sections 5.2.1.3 and 5.2.2.3 indicate that the IBER is well below the 10^{-2} bit error probability limit required at IEC 61508 SIL 3 for the 110 kbps and 850 kbps data rates. The IBER at the 6.81 Mbps data rate, however, exceeds the 10^{-2} bit error probability limit in most cases at the same distance. In both the office and industrial environments UWB would be able to operate at IEC 61508 SIL 3 as part of the black channel with a safety protocol such as PROFIsafe (see Section 2.2.3).

6.1 Future Work

UWB radio is capable of precision ranging to an accuracy of ±10 cm and is investigated in [51] using the DecaWave DW1000 tranceiver. Future work could consider combining the ranging and communication capabilities of UWB to permit precise positioning of motes within a WPAN. What ranging accuracy can be achieved, and would this affect communication within the WPAN? Could positioning be combined with voice traffic for localization of handheld radios in an industrial environment?

The UWB physical layer as defined in IEEE 802.15.4-2011 supports four data rates ranging from 110 kbps to 27.24 Mbps. Furthermore, the length of the preamble is also configurable. UWB configuration affects the bit error performance of the communication, but also allows users to tune the network for different characteristics such as latency or power consumption. Which configuration parameters provide the best latency or power consumption? Could these parameters be dynamically updated during the network operations to trade off performance and power consumption during varying environmental conditions?

As mentioned in Section 5.4.2, a data rate of 850 Kbps was used for UWB testing to accommodate the 127 byte data packets in the OpenWSN time slot of 15 ms. How much will UWB multi-hop network reliability improve if the data rate is reduced to 110 Kbps? How can OpenWSN be adapted to achieve this lower rate with DW1000 or other UWB radios? Are other IEEE 802.15.4 wireless process control MAC layer standards such as DSME and LLDN better able to take advantage of UWB communication properties?

References

[1] *Devices Using Ultra-Wideband (UWB) Technology*, Tech. report, Industry Canada, March 2009.

[2] *LTC5800-WHM Datasheet*, `http://cds.linear.com/docs/en/datasheet/5800whmfa.pdf`, 2013, Accessed 2017-02-26.

[3] *IEEE Standard for Low-Rate Wireless Networks*, IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011) (2016), 1–709.

[4] *IEEE Standard for Low-Rate Wireless Networks – Amendment 2: Ultra-Low Power Physical Layer*, IEEE Std 802.15.4q-2016 (Amendment to IEEE Std 802.15.4-2015 as amended by IEEE Std 802.15.4n-2016) (2016), 1–52.

[5] Omid Abedi and Mustapha CE Yagoub, *Performance comparison of uwb pulse modulation schemes under white gaussian noise channels*, International Journal of Microwave Science and Technology **2012** (2012).

[6] Z. Ahmadian and L. Lampe, *Performance analysis of the ieee 802.15.4a uwb system*, IEEE Transactions on Communications **57** (2009), no. 5, 1474–1485.

[7] J. Åkerberg, F. Reichenbach, and M. Björkman, *Enabling safety-critical wireless communication using WirelessHART and PROFIsafe*, 2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA) (Bilbao, Spain), Sept 2010, pp. 1–8.

[8] J. Åkerberg, F. Reichenbach, M. Gidlund, and M. Björkman, *Measurements on an industrial wireless HART network supporting PROFIsafe: A case study*, 2011 IEEE 16th Conference on Emerging Technologies Factory Automation (ETFA) (Toulouse, France), Sept 2011, pp. 1–8.

[9] Arduino AG, *Arduino Due*, `https://www.arduino.cc/en/Main/arduinoBoardDue`, Accessed 2017-02-20.

[10] Richard Candell, Kate A. Remley, Jeanne T. Quimby, David Novotny, Alexandra Curtin, Peter B. Papazian, Mohamed Kashef, and Joseph Diener, *Industrial wireless systems radio propagation measurements*, Tech. Report 1951, NIST, January 2017.

[11] Deji Chen, Mark Nixon, Song Han, Aloysius K Mok, and Xiuming Zhu, *WirelessHART and IEEE 802.15.4e*, 2014 IEEE International Conference on Industrial Technology (ICIT) (Busan, Korea), IEEE, 2014, pp. 760–765.

[12] J. Chen, X. Cao, P. Cheng, Y. Xiao, and Y. Sun, *Distributed collaborative control for industrial automation with wireless sensor and actuator networks*, IEEE Transactions on Industrial Electronics **57** (2010), no. 12, 4219–4230.

[13] Federal Communication Commission, *First report and order 02-48, part 15 (radio frequency devices), subpart F (ultra-wideband operation)*, Avaialble at `http://www.ecfr.gov/cgi-bin/text-idx?SID=e196d8602649db11750c5515e4b40438&mc=true&node=pt47.1.15&rgn=div5`, Accessed 2016-11-18.

[14] DecaWave, *DecaWave Products Overview*, `http://www.decawave.com/products/overview`, Accessed 2017-02-11.

[15] DecaWave, *APH010 DW1000 Inter-Channel Interference*, Tech. report, 2014, v0.2.

[16] DecaWave, *DW1000 Datasheet*, December 2015, Version 1.3.

[17] DecaWave, *DW1000 User Manual*, December 2015, Version 2.07.

[18] Neelam Dewangan, *A detailed Study of 4G in Wireless Communication: Looking insight in issues in OFDM*, Anchor Academic Publishing (aap_verlag), 2013.

[19] Dust Networks, *SmartMesh SDK*, `https://github.com/dustcloud/smartmeshsdk`, Accessed 2017-02-20.

[20] Alessandra Flammini and Emiliano Sisinni, *WirelessHART*, Industrial communication technology handbook, CRC Press, 2014, pp. 31–1 – 31–20.

[21] Lijia Ge, Guangrong Yue, and Sofiene Affes, *On the BER performance of pulse-position-modulation UWB radio in multipath channels*, 2002 IEEE Conference on Ultra Wideband Systems and Technologies, 2002. Digest of Papers, IEEE, 2002, pp. 231–234.

[22] Vikas Goyal and BS Dhaliwal, *Ultra wideband modulation performance improvement and advantages over conventional narrowband systems.*, Computer Science & Telecommunications **48** (2016), no. 2.

[23] V. C. Gungor, B. Lu, and G. P. Hancke, *Opportunities and challenges of wireless sensor networks in smart grid*, IEEE Transactions on Industrial Electronics **57** (2010), no. 10, 3557–3564.

[24] Gerhard P Hancke and Ben Allen, *Ultrawideband as an industrial wireless solution*, IEEE Pervasive Computing **5** (2006), no. 4, 78–85.

[25] Sinan Gezici I. G. Zafer Sahinoglu, *Ultra-wideband positioning systems: Theoretical limits, ranging algorithms, and protocols*, Cambridge University Press, October 2008.

[26] IEC, *Functional safety of electrical/electronic/programmable electronic safety-related systems*, IEC 61508:2010, Industrial Electromechanical Commission, Geneva, Switzerland, 2010.

[27] _____, *Industrial communication networks – profiles. Part 3, Functional Safety fieldbuses–general rules and profile definitions*, IEC 61784-3:2010, Industrial Electromechanical Commission, Geneva, Switzerland, 2010.

[28] _____, *Industrial communication networks–profiles. Part 3-3, Functional Safety fieldbuses – additional specifications for CPF 3*, IEC 61784-3-3:2010, Industrial Electromechanical Commission, Geneva, Switzerland, 2010.

[29] _____, *Industrial networks – Wireless communication network and communication profiles – WirelessHART*, IEC 62591:2016, Industrial Electromechanical Commission, Geneva, Switzerland, 2016.

[30] IEEE Computer Society, *IEEE Std 802.15.4e-2012 - IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer*, April 2012.

[31] Daniel M. King and Bradford G. Nickerson, *Ultra-Wideband Wireless Communication for Real-Time Control*, Tech. Report TR16-238, University of New Brunswick, May 2016.

[32] James F. Kurose and Keith W. Ross, *Computer Networking: A top down approach*, sixth ed., Pearson, 2013.

[33] Tomas Lennvall, Jan-Erik Frey, and Mikael Gidlund, *Wireless Sensor Networks for Automation*, Industrial communication technology handbook, CRC Press, 2014, pp. 36–1 – 36–51.

[34] Wasim Q Malik, Christopher J Stevens, and David J Edwards, *Multipath effects in ultrawideband rake reception*, IEEE Transactions on Antennas and Propagation **56** (2008), no. 2, 507–514.

[35] Robert McGill, John W Tukey, and Wayne A Larsen, *Variations of box plots*, The American Statistician **32** (1978), no. 1, 12–16.

[36] Dragan Mitić, Aleksandar Lebl, and Žarko Markov, *Calculating the required number of bits in the function of confidence level and error probability estimation*, Serbian Journal of Electrical Engineering **9** (2012), no. 3, 361–375.

[37] Guido Moritz and Frank Golatowski, *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) and Constrained Application Protocol (CoAP)*, Industrial communication technology handbook, CRC Press, 2014, pp. 38–1 – 38–13.

[38] G. Patti, G. Alderisi, and L. L. Bello, *Introducing multi-level communication in the IEEE 802.15.4e protocol: The MultiChannel-LLDN*, Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), Sept 2014, pp. 1–8.

[39] S. Petersen and S. Carlsen, *WirelessHART Versus ISA100.11a: The Format War Hits the Factory Floor*, IEEE Industrial Electronics Magazine **5** (2011), no. 4, 23–34.

[40] Stig Petersen and Niels Aakvaag, *Wireless instrumentation for safety critical systems*, Tech. Report A26762, SINTEF, March 2015, 50 pages.

[41] Stig Petersen and Simon Carlsen, *ISA100.11a*, Industrial communication technology handbook, CRC Press, 2014, pp. 32–1 – 32–14.

[42] Victoria Pimentel, *Estimating the safety function response time for wireless control systems*, Master's thesis, University of New Brunswick, 2015.

[43] Justin Redd, *Calculating statistical confidence levels for error-probability estimates*, Lightwave Magazine **21** (2000), no. 5, 110–114.

[44] Thomas Warren Rondeau and Charles W Bostian, *Artificial intelligence in wireless communications*, Artech House, 2009.

[45] Wolfgang Stripf and Herbert Barthel, *PROFIsafe: Functional Safety with PROFIBUS and PROFINET*, Industrial communication technology handbook, CRC Press, 2014, pp. 27–1 – 27–22.

[46] Peter Sweeney, *Error control coding from theory to practice*, John Wiley & Sons, 2002.

[47] P Thubert, A Brandt, J Hui, R Kelsey, P Levis, K Pister, R Struik, JP Vasseur, and R Alexander, *RPL: IPv6 routing protocol for low power and lossy networks*, RFC 6550 (2012).

[48] Thomas Watteyne, Xavier Vilajosana, Branko Kerkez, Fabien Chraim, Kevin Weekly, Qin Wang, Steven Glaser, and Kris Pister, *OpenWSN: a standards-based low-power wireless development environment*, Transactions on Emerging Telecommunications Technologies **23** (2012), no. 5, 480–493.

[49] Stephen B Wicker and Vijay K Bhargava, *Reed-Solomon codes and their applications*, John Wiley & Sons, 1999.

[50] Feng Xia, Yu-Chu Tian, Yanjun Li, and Youxian Sung, *Wireless sensor/actuator network design for mobile control applications*, Sensors **7** (2007), no. 10, 2157–2173.

[51] Mohammadreza Yavari, *Indoor real-time positioning using ultra-wideband technology*, Master's thesis, University of New Brunswick, 2015.

[52] W. Zeng, H. Wang, H. Yu, and A. Xu, *The Research and Application of UWB Based Industrial Network*, 2006 3rd International Conference on Ultrawideband and Ultrashort Impulse Signals (Sevastopol, Ukraine), Sept 2006, pp. 153–155.

[53] Li Zhao and A. M. Haimovich, *Performance of ultra-wideband communications in the presence of interference*, IEEE Journal on Selected Areas in Communications **20** (2002), no. 9, 1684–1691.

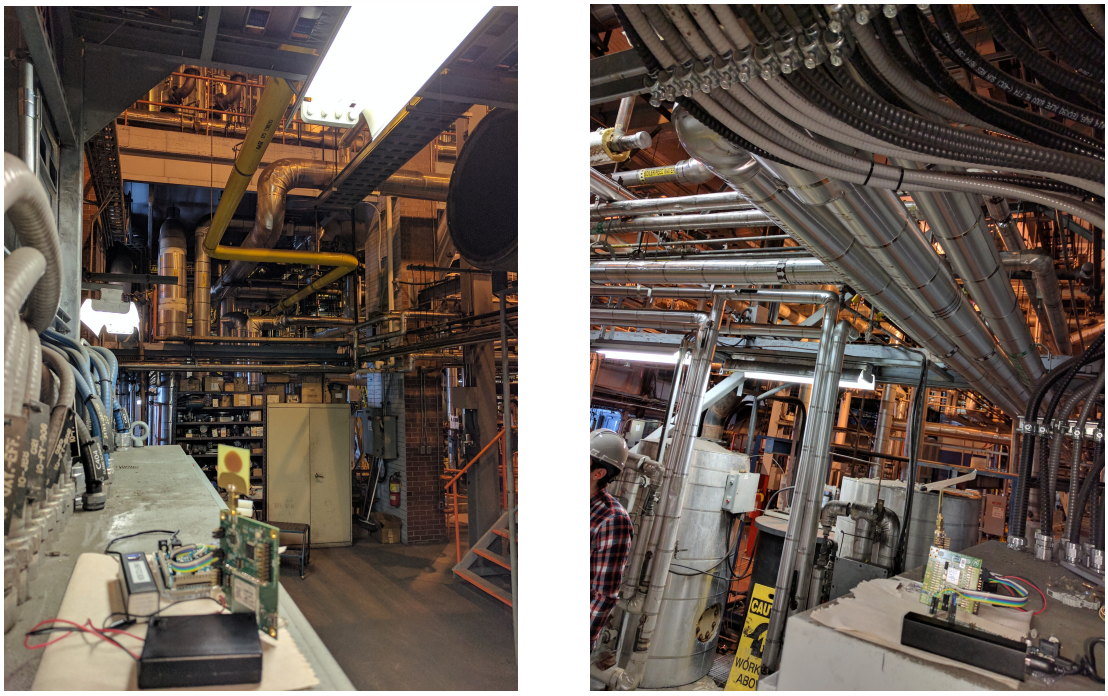
Appendix A

Mote Placement

The figures in this appendix show the placement of each mote at the UNB heating plant. The photographs were taken on January 31, 2017 and February 22, 2017.

(a) (b)

Figure A.1: Mote placement at the UNB heating plant for (a) DAG root and network manager, and (b) mote R1.



(a) (b)

Figure A.2: Mote placement at the UNB heating plant for (a) mote R2, and (b) mote R3.

(a) (b)

Figure A.3: Mote placement at the UNB heating plant for (a) mote R4, and (b) mote R5.

Appendix B

`uperiodicpush` Application

The `uperiodicpush` application runs as part of the OpenWSN firmware and is responsible for periodically generating upstream packets. The generation of upstream packets is triggered upon the reception of a request packet on UDP port 2005. A 2 second timer is then started which transmits a packet every two seconds. Each packet contains the following information:

1. A monotonically increasing sequence number.
2. The current ASN in which the packet is generated.
3. The MAC address of the mote.

```
#include "opendefs.h"
#include "urttlatency.h"
#include "openudp.h"
#include "openqueue.h"
#include "openserial.h"
#include "opentimers.h"
#include "scheduler.h"
#include "packetfunctions.h"
#include "IEEE802154E.h"
#include "idmanager.h"
#include <stdbool.h>

//=========================== variables ================================
```

```
#define REQUEST_LENGTH (2)
#define PACKET_LENGTH (2U + sizeof(asn_t) + 8U)
#define PUSH_TIMER_PERIOD_MS (2000)

typedef struct {
    open_addr_t target_addr;
    uint16_t target_port;
    uint16_t seq_num;
    uint16_t packets_remaining;
    bool is_active;
    opentimer_id_t timer_id;
} uperiodicpush_vars_t;

static uperiodicpush_vars_t app_vars;

//========================== prototypes ==============================
static void uperiodicpush_timer_cb(opentimer_id_t timer);
static void uperiodicpush_task_cb(void);

//========================== public ==================================

void uperiodicpush_init() {
    app_vars.is_active = FALSE;
}

void uperiodicpush_receive(OpenQueueEntry_t* request) {

    if ((request->length == REQUEST_LENGTH) && (!app_vars.is_active)) {
        app_vars.is_active         = TRUE;
        app_vars.packets_remaining =
            (uint16_t)request->payload[0] |
            (uint16_t)((uint16_t)request->payload[1] << 8U);
        app_vars.seq_num           = 0U;
        app_vars.target_port       = request->l4_sourcePortORicmpv6Type;
        app_vars.target_addr.type  = ADDR_128B;
        memcpy(app_vars.target_addr.addr_128b,
               request->l3_sourceAdd.addr_128b,
               16U);

        app_vars.timer_id = opentimers_start(PUSH_TIMER_PERIOD_MS,
                                             TIMER_PERIODIC,
                                             TIME_MS,
                                             &uperiodicpush_timer_cb);
    }

    openqueue_freePacketBuffer(request);
}
```

```
void uperiodicpush_sendDone(OpenQueueEntry_t* msg, owerror_t error) {
   openqueue_freePacketBuffer(msg);
}

static void uperiodicpush_timer_cb(opentimer_id_t timer) {
    scheduler_push_task(uperiodicpush_task_cb,TASKPRIO_COAP);
}

static void uperiodicpush_task_cb() {
    OpenQueueEntry_t* msg;
    uint8_t current_asn[sizeof(asn_t)];

    if ((app_vars.is_active) && (app_vars.packets_remaining > 0U)) {
        msg = openqueue_getFreePacketBuffer(COMPONENT_UPERIODICPUSH);

        if (msg == NULL) {
            return;
        }

        msg->owner                          = COMPONENT_UPERIODICPUSH;
        msg->creator                        = COMPONENT_UPERIODICPUSH;
        msg->l4_protocol                    = IANA_UDP;
        msg->l4_sourcePortORicmpv6Type      = WKP_UDP_PERIODICPUSH;
        msg->l4_destination_port            = app_vars.target_port;
        msg->l3_destinationAdd.type         = ADDR_128B;
        memcpy(msg->l3_destinationAdd.addr_128b,
               app_vars.target_addr.addr_128b,
               16U);

        ieee154e_getAsn(current_asn);

        packetfunctions_reserveHeaderSize(msg, PACKET_LENGTH);

        /* Add sequence number */
        msg->payload[0] = (uint8_t)(app_vars.seq_num & 0xFFU);
        msg->payload[1] = (uint8_t)(app_vars.seq_num >> 8U);

        /* Add current ASN */
        memcpy(&msg->payload[2], current_asn, sizeof(asn_t));

        /* Add my MAC address */
        memcpy(&msg->payload[2U + sizeof(asn_t)],
               idmanager_getMyID(ADDR_64B)->addr_64b,
               8U);

        if (E_SUCCESS == openudp_send(msg)) {
            app_vars.seq_num++;
            app_vars.packets_remaining--;
```

```
            if (app_vars.packets_remaining == 0U) {
                app_vars.is_active = FALSE;
                opentimers_stop(app_vars.timer_id);
            }
        }
        else {
            openqueue_freePacketBuffer(msg);
        }
    }
    else {
        app_vars.is_active = FALSE;
        opentimers_stop(app_vars.timer_id);
    }
}
```

Appendix C

`urttlatency` Application

The `urttlatency` application runs as part of the OpenWSN firmware and is responsible for generating reply packets to round trip request packets. The application operates similarly to a "ping" request, except additional ASN information is appended to the packet to permit the packet latency to be calculated.

```
#include "opendefs.h"
#include "urttlatency.h"
#include "openudp.h"
#include "openqueue.h"
#include "openserial.h"
#include "packetfunctions.h"
#include "IEEE802154E.h"
#include "idmanager.h"

//=========================== variables =======================================

#define REQUEST_LENGTH (2U)

#define RESPONSE_LENGTH ((sizeof(asn_t) * 2U) + 8U)

//=========================== prototypes ======================================

//=========================== public ==========================================

void urttlatency_init() {
}

void urttlatency_receive(OpenQueueEntry_t* request) {
```

```
OpenQueueEntry_t* reply;
uint8_t current_asn[sizeof(asn_t)];

reply = openqueue_getFreePacketBuffer(COMPONENT_URTTLATENCY);
if (reply==NULL) {
   openserial_printError(
      COMPONENT_URTTLATENCY,
      ERR_NO_FREE_PACKET_BUFFER,
      (errorparameter_t)0,
      (errorparameter_t)0
   );
   openqueue_freePacketBuffer(request);
   return;
}

reply->owner                     = COMPONENT_URTTLATENCY;
reply->creator                   = COMPONENT_URTTLATENCY;
reply->l4_protocol               = IANA_UDP;
reply->l4_destination_port       = request->l4_sourcePortORicmpv6Type;
reply->l4_sourcePortORicmpv6Type = request->l4_destination_port;
reply->l3_destinationAdd.type    = ADDR_128B;

// copy source to destination to echo.
memcpy(&reply->l3_destinationAdd.addr_128b[0],
       &request->l3_sourceAdd.addr_128b[0],
       16);

packetfunctions_reserveHeaderSize
    (reply,request->length + RESPONSE_LENGTH);

/* Copy request payload */
memcpy(&reply->payload[0],&request->payload[0],request->length);

/* Add the ASN when the request was received */
memcpy(&reply->payload[request->length],
       &request->l2_asn,
       sizeof(asn_t));

/* Add the ASN of the time at which this response is generated */
ieee154e_getAsn(current_asn);
memcpy(&reply->payload[request->length + sizeof(asn_t)],
       &current_asn,
       sizeof(asn_t));

/* Add my MAC address */
memcpy(&reply->payload[request->length + (sizeof(asn_t) * 2U)],
       idmanager_getMyID(ADDR_64B)->addr_64b,
       8U);
```

```
   openqueue_freePacketBuffer(request);

   if ((openudp_send(reply))==E_FAIL) {
      openqueue_freePacketBuffer(reply);
   }
}

void urttlatency_sendDone(OpenQueueEntry_t* msg, owerror_t error) {
   openqueue_freePacketBuffer(msg);
}
```

Appendix D

OpenVisualizer Latency Measurement

Packets received at the DAG root mote are forwarded to OpenVisualizer through a serial connection. The DAG root also appends the ASN in which the packet was received which is used to calculate the upstream latency. We have modified OpenVisualizer's ParserData class to capture packets generated by the urttlatency and uperiodicpush applications. The relevant information from the packet is saved in a unique file based on the mote's MAC address. OpenVisualizer captures the packets by inspecting the source port of the UDP packet to determine from which application the packet originated.

```
    def parseInput(self,input):
        # log
        if log.isEnabledFor(logging.DEBUG):
            log.debug("received data {0}".format(input))

        # ensure input not short longer than header
        self._checkLength(input)

        headerBytes = input[:2]
        #asn comes in the next 5bytes.

        asnbytes=input[2:7]
```

```
(self._asn) = struct.unpack('<BHH','').join([chr(c) for c in asnbytes]))

#source and destination of the message
dest = input[7:15]

#source is elided!!! so it is not there.. check that.
source = input[15:23]

# remove asn src and dest and mote id at the beginning.
# this is a hack for latency measurements...
# TODO, move latency to an app listening on the corresponding port.
# inject end_asn into the packet as well
input = input[23:]

if log.isEnabledFor(logging.DEBUG):
    log.debug("packet without source,dest and asn {0}".format(input))

# when the packet goes to internet it comes
# with the asn at the beginning as timestamp.

if len(input) >37:
   if input[-36] == 0x07 and input[-35] == 0xd2:
       # udp port 2002 for urttlatency app
       data = input[-28:]

       timestamp   = struct.unpack(
           '<Q',
           ''.join(chr(x) for x in data[-28:-20])
       )[0]
       req_seqnum  = struct.unpack(
           '<H',
           ''.join(chr(x) for x in data[-20:-18])
       )[0]
       req_rx_asn  = asntonumFromStruct(data[-18:-13])
       resp_tx_asn = asntonumFromBytes(data[-13:-8])
       resp_rx_asn = asntonumFromBytes(asnbytes)
       mac_addr    = data[-8:]

       mac_addr = '-'.join('{:02x}'.format(x) for x in mac_addr)

       timenow = currentTimeMillis()

       f = open('rtt_' + mac_addr + '.csv', 'a')
       f.write('{0},{1},{2},{3},{4},{5}\n'.format(
           timenow,
           timestamp,
           req_seqnum,
           req_rx_asn,
```

```
            resp_tx_asn,
            resp_rx_asn,
        ))
        f.close()
    elif input[-23] == 0x07 and input[-22] == 0xd5:
        # udp port 2005 for uperiodicpush app
        data = input[-28:]

        seqnum   = struct.unpack(
            '<H',
            ''.join(chr(x) for x in data[-15:-13])
        )[0]
        tx_asn   = asntonumFromBytes(data[-13:-8])
        rx_asn   = asntonumFromBytes(asnbytes)
        mac_addr = data[-8:]

        mac_addr = '-'.join('{:02x}'.format(x) for x in mac_addr)

        f = open('upstream_' + mac_addr + '.csv', 'a')
        f.write('{0},{1},{2}\n'.format(
            seqnum,
            tx_asn,
            rx_asn,
        ))
        f.close()
else:
    print input
    pass

eventType='data'
# notify a tuple including source as one hop away
# nodes elide SRC address as can be inferred from MAC layer header
return eventType, (source, input)
```

Appendix E

`unbrinfo` Application

The `unbrinfo` application runs as part of the OpenWSN firmware and is responsible for returning information about neighboring motes. The application listens for UDP requests and replies with the requested row from the mote's neighbor table. The purpose of this application is to permit applications running on the DAG root to extract the neighbor information from each mote wirelessly.

```
#include "opendefs.h"
#include "unbrinfo.h"
#include "openudp.h"
#include "openqueue.h"
#include "openserial.h"
#include "packetfunctions.h"
#include "IEEE802154E.h"
#include "idmanager.h"

void unbrinfo_init() {
}

void unbrinfo_receive(OpenQueueEntry_t* request) {
   OpenQueueEntry_t* reply;
   neighborRow_t row;
   uint8_t current_asn[sizeof(asn_t)];

   if (request->length != 1U) {
       openqueue_freePacketBuffer(request);

   } else {
```

```
/* The request payload contains the
 * index of the neighbor row to read
 */
if (neighbors_getRow(request->payload[0], &row)) {
    reply = openqueue_getFreePacketBuffer(COMPONENT_UNBRINFO);
    if (reply==NULL) {
       openserial_printError(
          COMPONENT_URTTLATENCY,
          ERR_NO_FREE_PACKET_BUFFER,
          (errorparameter_t)0,
          (errorparameter_t)0
       );
       openqueue_freePacketBuffer(request);
       return;
    }

    reply->owner                     = COMPONENT_UNBRINFO;
    reply->creator                   = COMPONENT_UNBRINFO;
    reply->l4_protocol               = IANA_UDP;
    reply->l4_destination_port       =
        request->l4_sourcePortORicmpv6Type;
    reply->l4_sourcePortORicmpv6Type = request->l4_destination_port;
    reply->l3_destinationAdd.type    = ADDR_128B;

    // copy source to destination to echo.
    memcpy(&reply->l3_destinationAdd.addr_128b[0],
           &request->l3_sourceAdd.addr_128b[0],
           16);

    packetfunctions_reserveHeaderSize
        (reply, sizeof(neighborRow_t) + 1);

    /* Reply contains the index of the neighbor row,
     * and the contents of the neighbor entry.
     */
    reply->payload[0] = request->payload[0];
    memcpy(&reply->payload[1], &row, sizeof(neighborRow_t));

    openqueue_freePacketBuffer(request);

    if ((openudp_send(reply))==E_FAIL) {
       openqueue_freePacketBuffer(reply);
    }
}

else {
    openqueue_freePacketBuffer(request);
```

```
        }
    }
}

void unbrinfo_sendDone(OpenQueueEntry_t* msg, owerror_t error) {
    openqueue_freePacketBuffer(msg);
}
```

Appendix F

`uscheduleinfo` Application

The `uscheduleinfo` application runs as part of the OpenWSN firmware and is responsible for returning the mote's TSCH schedule. The application listens for UDP requests and replies with the requested row from the mote's schedule. The purpose of this application is to permit applications running on the DAG root to extract the network schedule information from each mote wirelessly.

```
#include "opendefs.h"
#include "unbrinfo.h"
#include "openudp.h"
#include "openqueue.h"
#include "openserial.h"
#include "packetfunctions.h"
#include "IEEE802154E.h"
#include "idmanager.h"
#include "schedule.h"

void uscheduleinfo_init() {
}

void uscheduleinfo_receive(OpenQueueEntry_t* request) {
   OpenQueueEntry_t* reply;
   debugScheduleEntry_t row;
   uint8_t current_asn[sizeof(asn_t)];

   if (request->length != 1U) {
       openqueue_freePacketBuffer(request);
```

```
} else {

    /* The request payload contains the index
     * of the schedule row to read
     */
    if (schedule_getScheduleEntry(request->payload[0], &row)) {
        reply = openqueue_getFreePacketBuffer(COMPONENT_USCHEDULEINFO);
        if (reply==NULL) {
           openserial_printError(
              COMPONENT_USCHEDULEINFO,
              ERR_NO_FREE_PACKET_BUFFER,
              (errorparameter_t)0,
              (errorparameter_t)0
           );
           openqueue_freePacketBuffer(request);
           return;
        }

        reply->owner                      = COMPONENT_USCHEDULEINFO;
        reply->creator                    = COMPONENT_USCHEDULEINFO;
        reply->l4_protocol                = IANA_UDP;
        reply->l4_destination_port        =
            request->l4_sourcePortORicmpv6Type;
        reply->l4_sourcePortORicmpv6Type = request->l4_destination_port;
        reply->l3_destinationAdd.type     = ADDR_128B;

        // copy source to destination to echo.
        memcpy(&reply->l3_destinationAdd.addr_128b[0],
               &request->l3_sourceAdd.addr_128b[0],
               16);

        packetfunctions_reserveHeaderSize
            (reply, sizeof(debugScheduleEntry_t) + 1);

        /* Reply contains the index of the schedule row,
         * and the contents of the schedule entry.
         */
        reply->payload[0] = request->payload[0];
        memcpy(&reply->payload[1], &row, sizeof(debugScheduleEntry_t));

        openqueue_freePacketBuffer(request);

        if ((openudp_send(reply))==E_FAIL) {
           openqueue_freePacketBuffer(reply);
        }
    }

    else {
```

```
            openqueue_freePacketBuffer(request);
        }
    }
}

void uscheduleinfo_sendDone(OpenQueueEntry_t* msg, owerror_t error) {
    openqueue_freePacketBuffer(msg);
}
```

Appendix G

rtt_test.py Application

The rtt_test.py application runs on the host computer and is responsible for running the round trip time measurements for the WirelessHART network. The application communicates with the WirelessHART network manager via the XML-RPC interface using the SmartMesh SDK. It is written in the Python programming language.

```
import argparse
import collections
import struct
import threading
import datetime, threading, time
from SmartMeshSDK.HartMgrConnector import HartMgrConnector

TEST_INTERVAL_SECS = 5

class NotifThread(threading.Thread):

    def __init__(self, macAddr):
        threading.Thread.__init__(self)
        self.data = []
        self.packetSent = []
        self.macAddr = macAddr

    def run(self):
        mgr = HartMgrConnector.HartMgrConnector()

        mgr.connect({'host':'192.168.99.100', 'port':4445})
```

```
        notif_token = mgr.dn_subscribe('data events')

        while True:
            notif = mgr.getNotification(60)

            if not notif:
                break
            else:
                if notif[0] == 'data' and notif[1].macAddr == self.macAddr:
                    payload = notif[1].payload
                    (sec, usec, reqcnt, seqnum) = struct.unpack(
                        '<IIII',
                        ''.join(chr(x) for x in payload[6:22])
                    )

                    mgrTime = int(mgr.dn_getTime().utc_time * 1000)

                    print "Got reply with seq={}".format(seqnum)

                    self.data.append({
                        'seq'    : seqnum,
                        't3'     : notif[1].time,
                        't4'     : int(mgr.dn_getTime().utc_time * 1000),
                        'reqcnt' : reqcnt
                    })

                elif notif[0] == 'PacketSent':
                    if notif[1].macAddr == self.macAddr:
                        self.packetSent.append({
                            'callbackId' : notif[1].callbackId,
                            't2'         : notif[1].timeStamp,
                        })

        mgr.disconnect()


def send_pings(mgr, moteMAC, npackets):
    packets_info = []

    try:
        for n in range(npackets):
            mgrTime = int(mgr.dn_getTime().utc_time * 1000)
            data = list(struct.pack('<IQ', n, mgrTime))
            data = map(ord,data)

            print "Sending request {0}/{1}...".format(n+1,npackets)
            cb = mgr.dn_sendRequest(
                moteMAC,
```

```
                'maintenance',
                'high',
                False,
                [0x00, 0x00, 0xFC, 0x12, 0x03] + data
            )

            packets_info.append({'seq'        : n,
                                 't1'         : mgrTime,
                                 't2'         : '',
                                 't3'         : '',
                                 't4'         : '',
                                 'reqcnt'     : '',
                                 'callbackId' : cb.callbackId})

            time.sleep(TEST_INTERVAL_SECS)
    except Exception as ex:
        print "Aborting due to exception:"
        print str(ex)
    except KeyboardInterrupt:
        print "Aborting due to keyboard interrupt"

    return packets_info


def combine_data(txPackets, packetSent, data):
    combined = []

    for txp in txPackets:

        # Find the corresponding PacketSent notification for this packet
        # Search by callbackId
        for ps in packetSent:
            if ps['callbackId'] == txp['callbackId']:
                txp['t2'] = ps['t2']
                break

        # Find the corresponding data notification for this packet
        # Search by packet sequence number
        for d in data:
            if d['seq'] == txp['seq']:
                txp['t3'] = d['t3']
                txp['t4'] = d['t4']
                txp['reqcnt'] = d['reqcnt']

        combined.append(txp)

    return combined
```

```
def write_output_file(packets_info, filename):
    f = open(filename, 'w')
    f.write('Req. Created Time')
    f.write(',Req. Tx Time')
    f.write(',Req. Rx Time')
    f.write(',Resp. Rx Time')
    f.write(',Num. Requests Received')
    f.write(',Sequence Number\n')

    for p in packets_info:
        f.write('{0},{1},{2},{3},{4},{5}\n'.format(
            p['t1'],
            p['t2'],
            p['t3'],
            p['t4'],
            p['reqcnt'],
            p['seq'],
        ))

    f.close()

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="WirelessHART RTT Tester")
    parser.add_argument(
        '--out',
        nargs=1,
        help='Output CSV file'
    )
    parser.add_argument(
        '--mac',
        nargs=1,
        help='MAC address of target mote'
    )
    parser.add_argument(
        '-N',
        nargs='?',
        type=int,
        default=60,
        help='Number of packets to send'
    )

    args = parser.parse_args()

    print "Starting notification thread"
    notifs = NotifThread(args.mac[0])
    notifs.start()
```

```
mgr = HartMgrConnector.HartMgrConnector()
mgr.connect({'host':'192.168.99.100', 'port':4445})
print "Connected"

# A small delay to allow the the NotifThread to connect.
time.sleep(1)

print "Sending requests to " + str(args.mac[0])
txPackets = send_pings(mgr, args.mac[0], args.N)

print "Waiting for remaining notifications..."
notifs.join()

print "Writing results to " + str(args.out[0])
packets_info = combine_data(txPackets, notifs.packetSent, notifs.data)

write_output_file(packets_info, args.out[0])
```

Appendix H

upstream_test.py Application

The upstream_test.py application runs on the host computer and is responsible for running the upstream latency measurements for the WirelessHART network. The application communicates with the WirelessHART network manager via the XML-RPC interface using the SmartMesh SDK. It is written in the Python programming language.

```
import argparse
import collections
import struct
import threading
import datetime, threading, time
from SmartMeshSDK.HartMgrConnector import HartMgrConnector

TEST_INTERVAL_SECS = 5

class NotifThread(threading.Thread):

    def __init__(self, macAddr):
        threading.Thread.__init__(self)
        self.data = []
        self.packetSent = []
        self.macAddr = macAddr

    def run(self):
        mgr = HartMgrConnector.HartMgrConnector()

        mgr.connect({'host':'192.168.99.100', 'port':4445})
```

```
        notif_token = mgr.dn_subscribe('data events')

        while True:
            notif = mgr.getNotification(60)

            if not notif:
                break
            else:
                if notif[0] == 'data' and notif[1].macAddr == self.macAddr:
                    payload = notif[1].payload
                    (sec, usec, reqcnt, seqnum) = struct.unpack(
                        '<IIII',
                        ''.join(chr(x) for x in payload[6:22])
                    )

                    mgrTime = int(mgr.dn_getTime().utc_time * 1000)

                    print "Got reply with seq={}".format(seqnum)

                    self.data.append({
                        'seq'    : seqnum,
                        't3'     : notif[1].time,
                        't4'     : int(mgr.dn_getTime().utc_time * 1000),
                        'reqcnt' : reqcnt
                    })

                elif notif[0] == 'PacketSent':
                    if notif[1].macAddr == self.macAddr:
                        self.packetSent.append({
                            'callbackId' : notif[1].callbackId,
                            't2'         : notif[1].timeStamp,
                        })

        mgr.disconnect()


def send_pings(mgr, moteMAC, npackets):
    packets_info = []

    try:
        for n in range(npackets):
            mgrTime = int(mgr.dn_getTime().utc_time * 1000)
            data = list(struct.pack('<IQ', n, mgrTime))
            data = map(ord,data)

            print "Sending request {0}/{1}...".format(n+1,npackets)
            cb = mgr.dn_sendRequest(
                moteMAC,
```

```
                'maintenance',
                'high',
                False,
                [0x00, 0x00, 0xFC, 0x12, 0x03] + data
            )

            packets_info.append({'seq'        : n,
                                 't1'         : mgrTime,
                                 't2'         : '',
                                 't3'         : '',
                                 't4'         : '',
                                 'reqcnt'     : '',
                                 'callbackId' : cb.callbackId})

            time.sleep(TEST_INTERVAL_SECS)
    except Exception as ex:
        print "Aborting due to exception:"
        print str(ex)
    except KeyboardInterrupt:
        print "Aborting due to keyboard interrupt"

    return packets_info


def combine_data(txPackets, packetSent, data):
    combined = []

    for txp in txPackets:

        # Find the corresponding PacketSent notification for this packet
        # Search by callbackId
        for ps in packetSent:
            if ps['callbackId'] == txp['callbackId']:
                txp['t2'] = ps['t2']
                break

        # Find the corresponding data notification for this packet
        # Search by packet sequence number
        for d in data:
            if d['seq'] == txp['seq']:
                txp['t3'] = d['t3']
                txp['t4'] = d['t4']
                txp['reqcnt'] = d['reqcnt']

        combined.append(txp)

    return combined
```

```
def write_output_file(packets_info, filename):
    f = open(filename, 'w')
    f.write('Req. Created Time')
    f.write(',Req. Tx Time')
    f.write(',Req. Rx Time')
    f.write(',Resp. Rx Time')
    f.write(',Num. Requests Received')
    f.write(',Sequence Number\n')

    for p in packets_info:
        f.write('{0},{1},{2},{3},{4},{5}\n'.format(
            p['t1'],
            p['t2'],
            p['t3'],
            p['t4'],
            p['reqcnt'],
            p['seq'],
        ))

    f.close()

if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="WirelessHART upstream Tester"
    )
    parser.add_argument(
        '--out',
        nargs=1,
        help='Output CSV file'
    )
    parser.add_argument(
        '--mac',
        nargs=1,
        help='MAC address of target mote'
    )
    parser.add_argument(
        '-N',
        nargs='?',
        type=int,
        default=60,
        help='Number of packets to send'
    )

    args = parser.parse_args()

    print "Starting notification thread"
    notifs = NotifThread(args.mac[0])
```

```
notifs.start()

mgr = HartMgrConnector.HartMgrConnector()
mgr.connect({'host':'192.168.99.100', 'port':4445})
print "Connected"

# A small delay to allow the the NotifThread to connect.
time.sleep(1)

print "Sending requests to " + str(args.mac[0])
txPackets = send_pings(mgr, args.mac[0], args.N)

print "Waiting for remaining notifications..."
notifs.join()

print "Writing results to " + str(args.out[0])
packets_info = combine_data(txPackets, notifs.packetSent, notifs.data)

write_output_file(packets_info, args.out[0])
```

Appendix I

RF Measurements

This appendix shows the RF measurements for all testing days during which tests were conducted for this thesis.

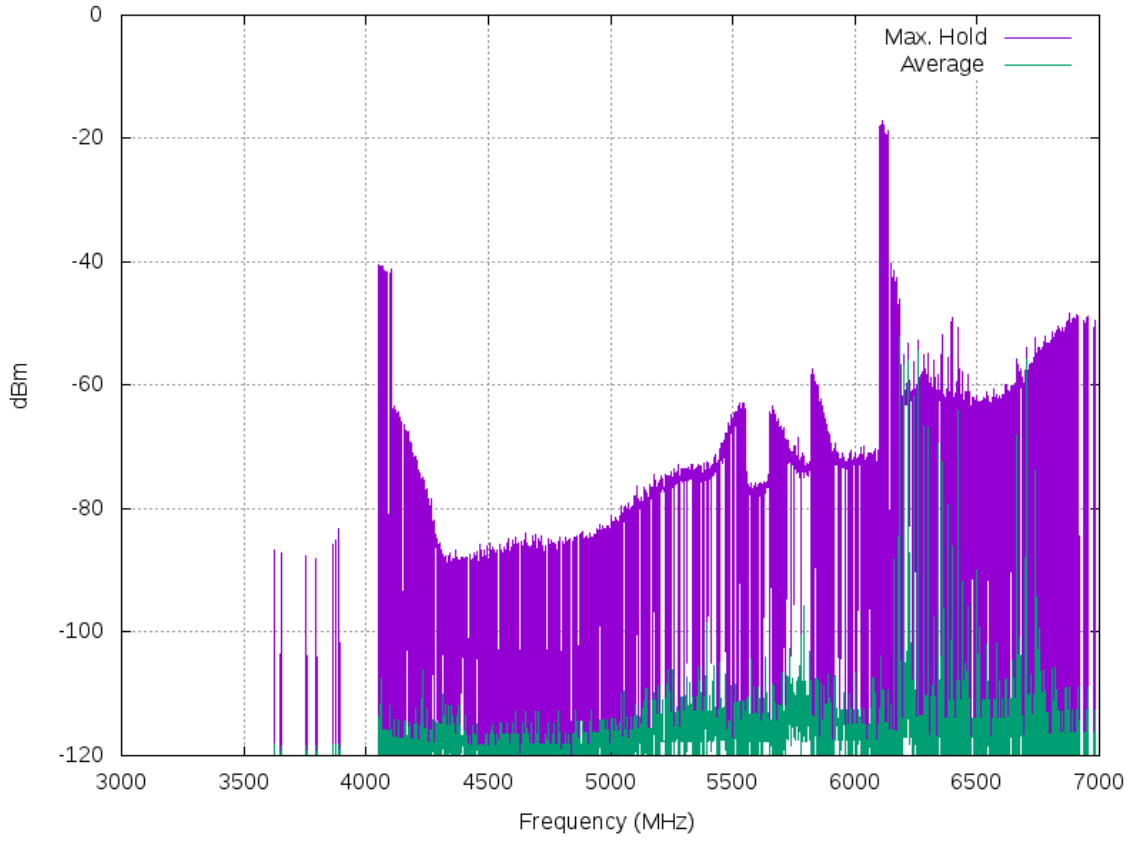


Figure I.1: Spectrum analysis at position T1 in the UNB ITC building on February 17, 2017 from 3 GHz to 7 GHz.

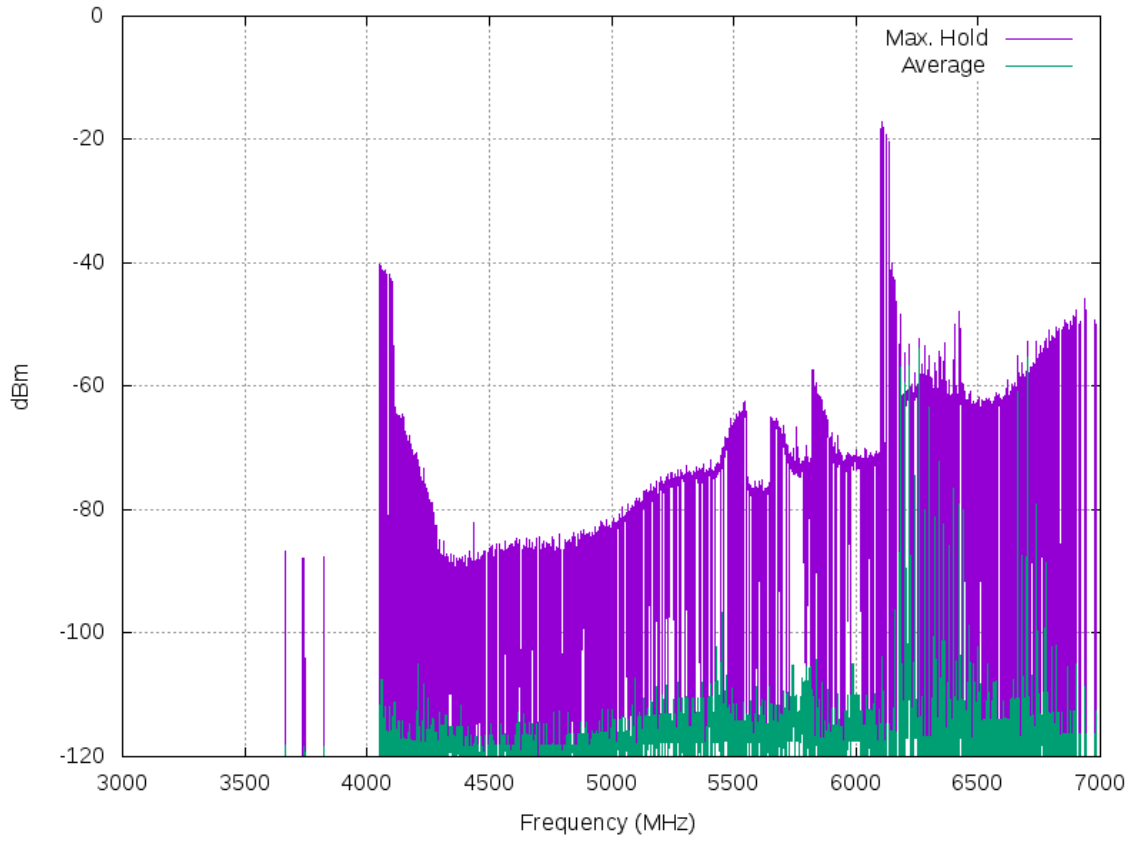


Figure I.2: Spectrum analysis at position T1 in the UNB ITC building on February 21, 2017 from 3 GHz to 7 GHz.

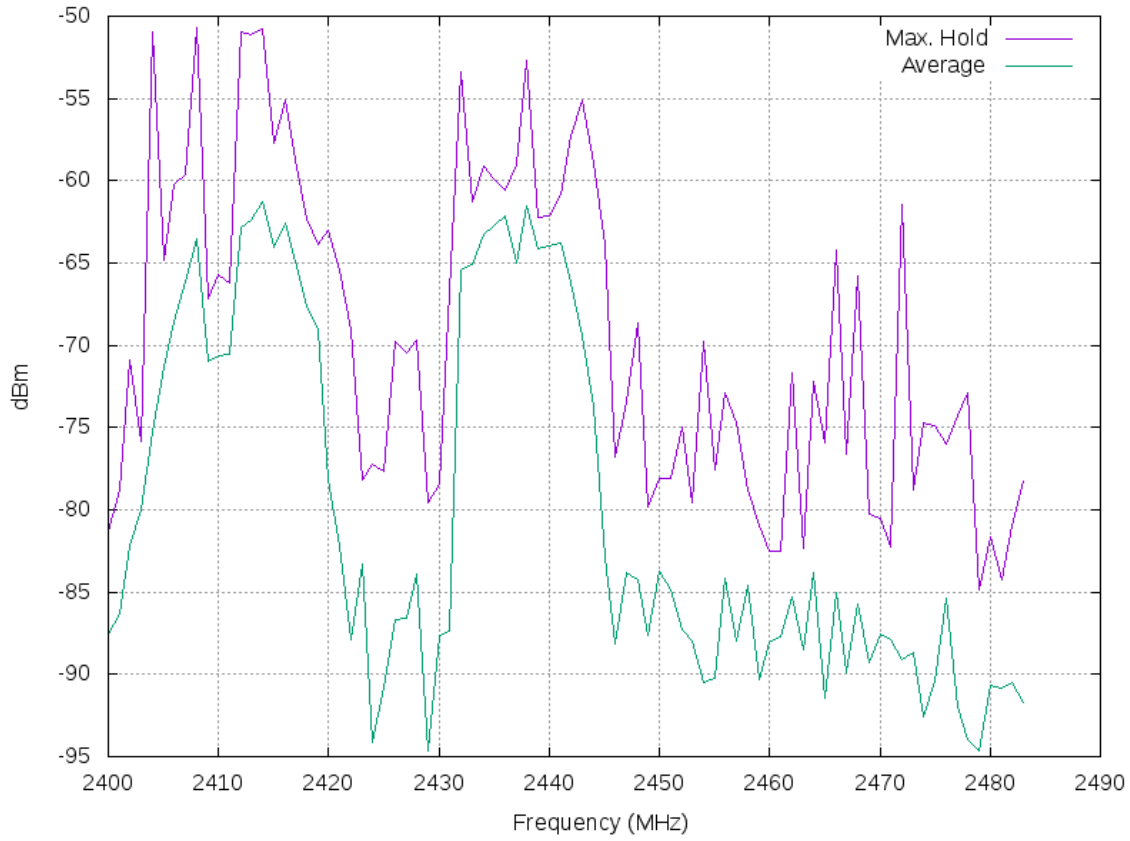


Figure I.3: Spectrum analysis at position T1 in the UNB ITC building on February 21, 2017 from 2.4 GHz to 2.483 GHz.

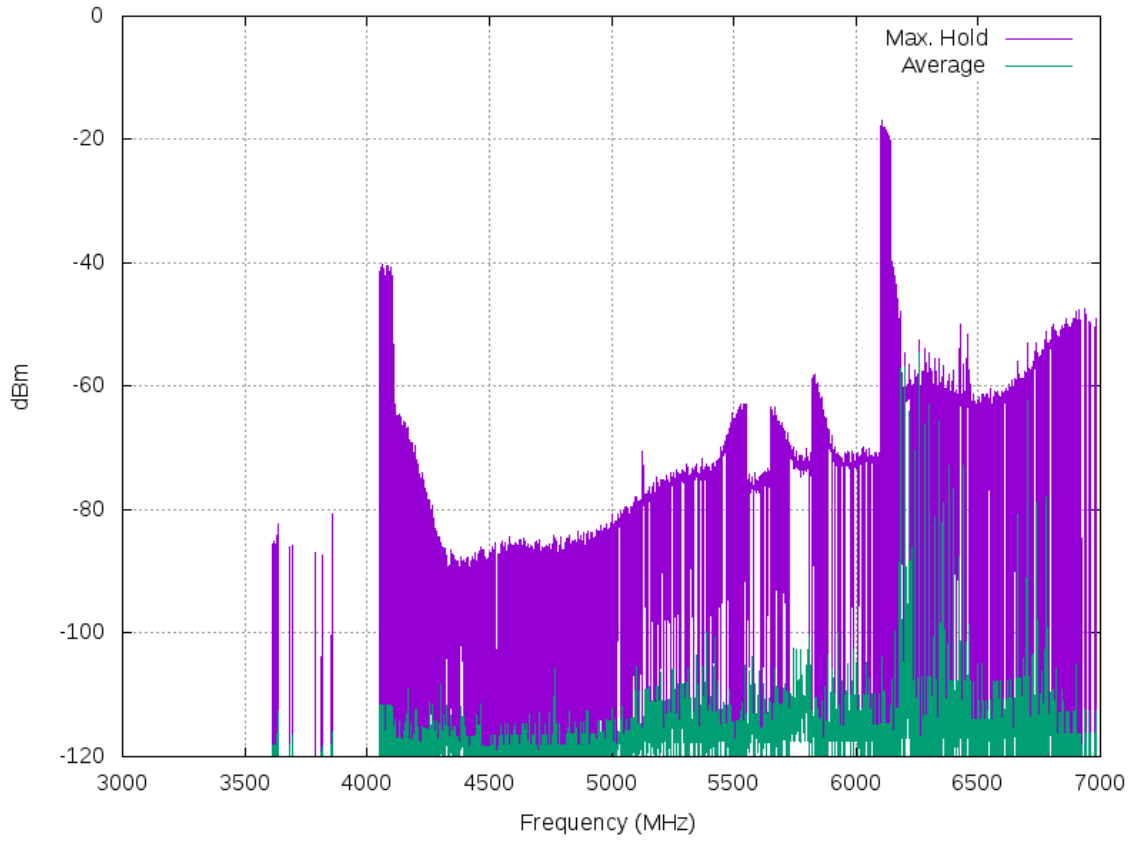


Figure I.4: Spectrum analysis at position T1 in the UNB heating plant on February 22, 2017 from 3 GHz to 7 GHz.

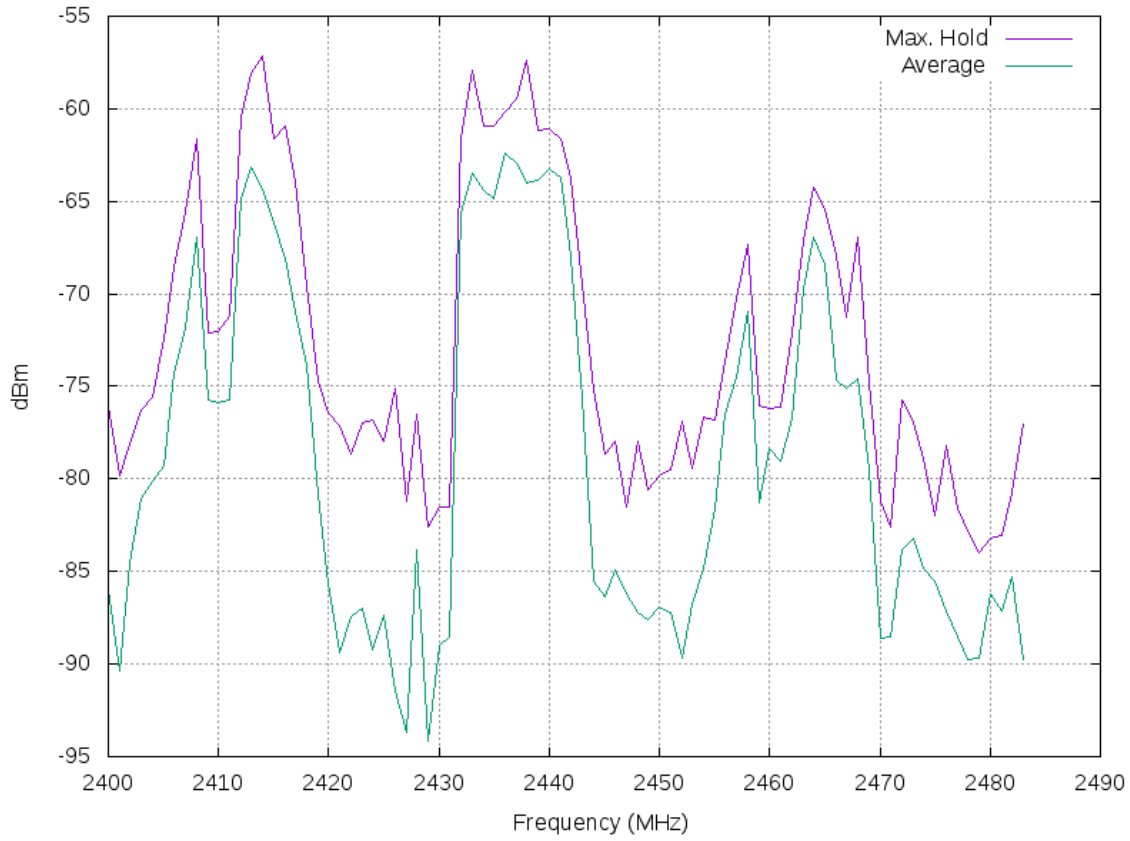


Figure I.5: Spectrum analysis at position T1 in the UNB heating plant on February 22, 2017 from 2.4 GHz to 2.483 GHz.

Appendix J

Network Reliability Issues

This appendix describes the problems that we encountered during initial testing using the OpenWSN and WirelessHART networks.

J.1 OpenWSN

Occasionally, during the joining process for a mote, the network schedule between the joining mote and the DAG root would become inconsistent where the DAG root allocates an extra slot in the schedule that the joining mote is not aware of. For example, the DAG root allocates 2 slots in the network schedule, but the joining mote is only aware of 1 of the slots or vice versa. This can result in degraded communication reliability due to a receiving mote not listening for packets during the inconsistent slot in the schedule.

The schedule inconsistency also caused the DAG root to run out of slots in its schedule, as the number of allocated schedule entries is limited due to RAM allocation. When this occurred, the OpenWSN detects the schedule overflow and reboots itself, causing the network to collapse. We found that the network schedule inconsistency problems could be mostly avoided by joining motes to the network one-by-one, instead of allowing all motes to join at the same time.

J.2 WirelessHART

We found that the WirelessHART network was more stable and reliable than the OpenWSN network, although two problems were encountered during testing. The first problem encountered was that a mote sometimes disconnected from the network during a test. This occurred on two occasions during testing at the UNB ITC building for the mote at position R4. This impacted the test as no further packets could be sent to or received from the disconnected mote, and any child motes in the network. The mote would usually rejoin the network after several minutes, but we were able to speed up rejoining by rebooting the mote, including our test firmware.

The second problem encountered was queuing delays due to the network manager's limitation on the downstream bandwidth. For our configuration, the network manager was able to send one downstream packet to each mote approximately every 2.5 seconds. Our first round trip tests sent packets at a rate of 1 packet every 3 seconds, which initially appeared to be sufficient. However, some delays in the network would occasionally cause packets to take longer than 2.5 seconds to be sent. Since the network manager sends packets in sequence to the mote, this causes the next packets to be queued at the network manager until the previous packets have finished sending. We observed during some test runs that the queuing delay would steadily increase for a period of time until the network manager's queue became full, at which point the test could not continue. Increasing the packet transmit rate to 1 packet every 5 seconds was sufficient to enable the manager to gradually recover from periods of extended queuing delays.

This queuing delay only affected communication in the downstream direction. Packets sent in the upstream service were allocated more bandwidth by the network manager and did not result in queuing delays.

Appendix K

OpenWSN Schedules

Tables K.1, K.2, and K.3 show the schedule used by OpenWSN during the upstream and round trip tests at the UNB ITC building, and both tests at the UNB heating plant respectively.

During the round trip test at the UNB ITC building the time slot at slot offset 9 and channel offset 7 was double-allocated to two motes R1 and R4. We suspect that this was a random occurrence which was not detected by OpenWSN as neither R1 nor R4 were on different halves of the network topology. However, this collision did not appear to affect the link stability as both motes reported 100% link stability for this time slot.

Channel Offset	Slot Offset										
	0	1	2	3	4	5	6	7	8	9	10
0	anycast										
1							$R4 \rightarrow R3$				
2								$R3 \rightarrow DR$		$R3 \rightarrow DR$	$R3 \rightarrow DR$
3									$R4 \rightarrow R3$		
4								$R1 \rightarrow M2$			
5					$R5 \rightarrow R4$						
6											
7					$R2 \rightarrow DR$				$R2 \rightarrow DR$		

Table K.1: OpenWSN schedule used during the upstream tests at the UNB ITC building.

Channel Offset	Slot Offset										
	0	1	2	3	4	5	6	7	8	9	10
0	anycast				$R5 \rightarrow R4$						
1							$R4 \rightarrow R3$				
2							$R2 \rightarrow DR$		$R3 \rightarrow DR$		
3											
4											
5											$R3 \rightarrow DR$
6											
7						$R2 \rightarrow DR$		$R3 \rightarrow DR$		$R1 \rightarrow R2$ $R4 \rightarrow R3$	

Table K.2: OpenWSN schedule used during the round trip tests at the UNB ITC building.

Channel Offset	Slot Offset										
	0	1	2	3	4	5	6	7	8	9	10
0	anycast										$R4 \rightarrow DR$
1						$R5 \rightarrow DR$	$R4 \rightarrow DR$			$R3 \rightarrow R4$	
2											
3									$R2 \rightarrow R5$		
4								$R1 \rightarrow R4$		$R5 \rightarrow DR$	
5											
6											
7					$R4 \rightarrow DR$						

Table K.3: OpenWSN schedule used during both tests at the UNB heating plant.

Appendix L

Latency Measurements

This appendix presents plots for the upstream and round trip latency measurements for each sample for all tests and locations.

L.1 Upstream Latency

L.1.1 UNB ITC Building

L.1.1.1 UWB

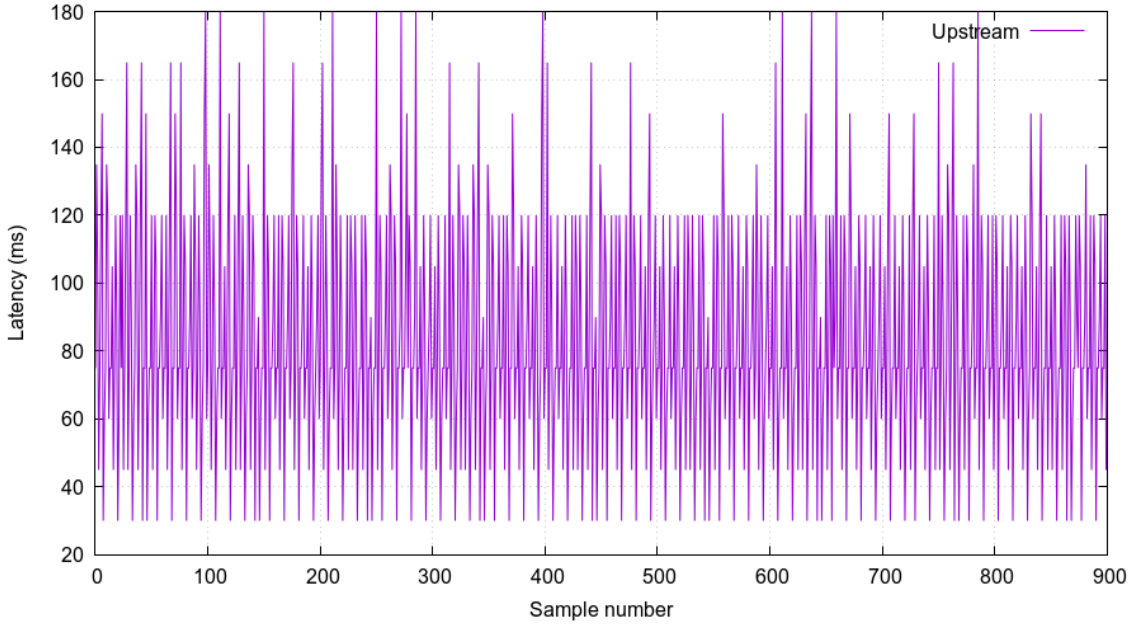


Figure L.1: UWB upstream latency for mote R1 at the UNB ITC building.

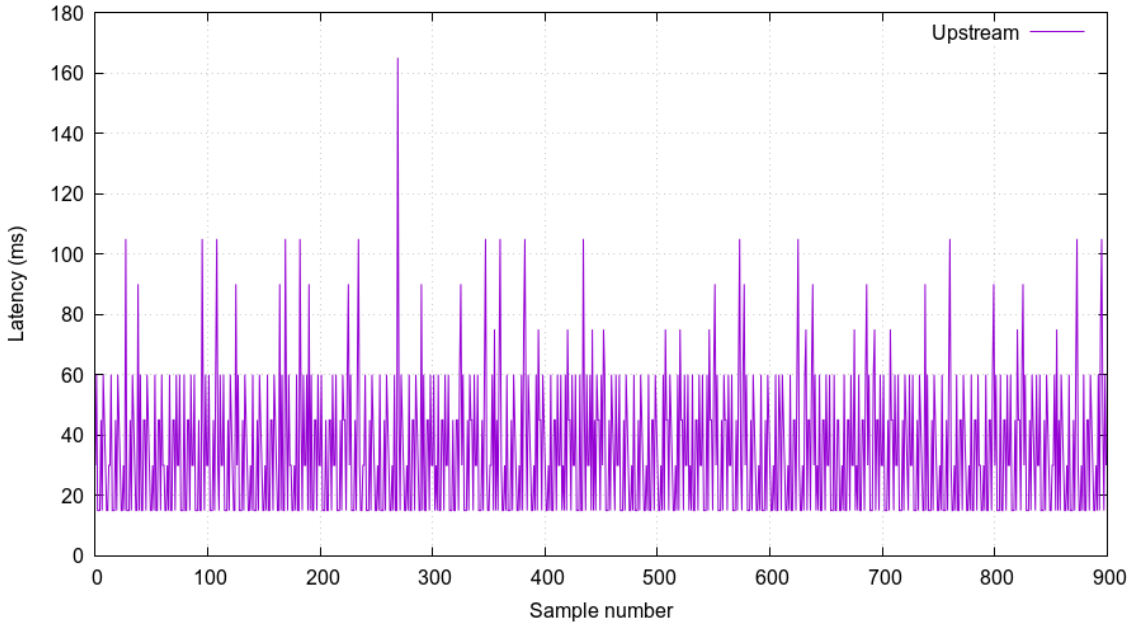


Figure L.2: UWB upstream latency for mote R2 at the UNB ITC building.

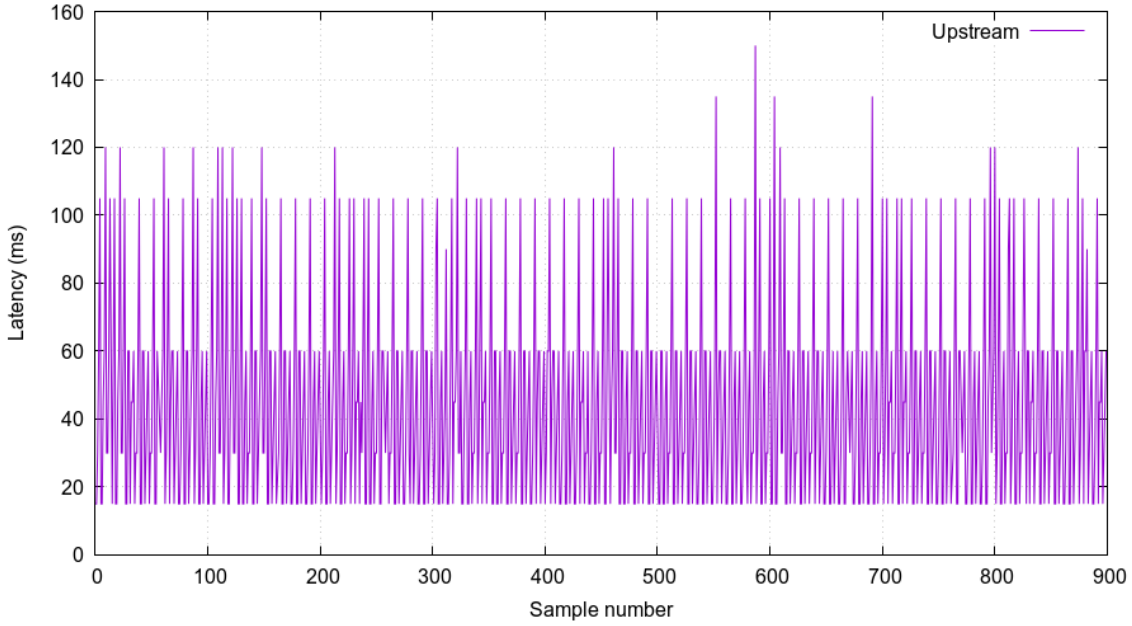


Figure L.3: UWB upstream latency for mote R3 at the UNB ITC building.

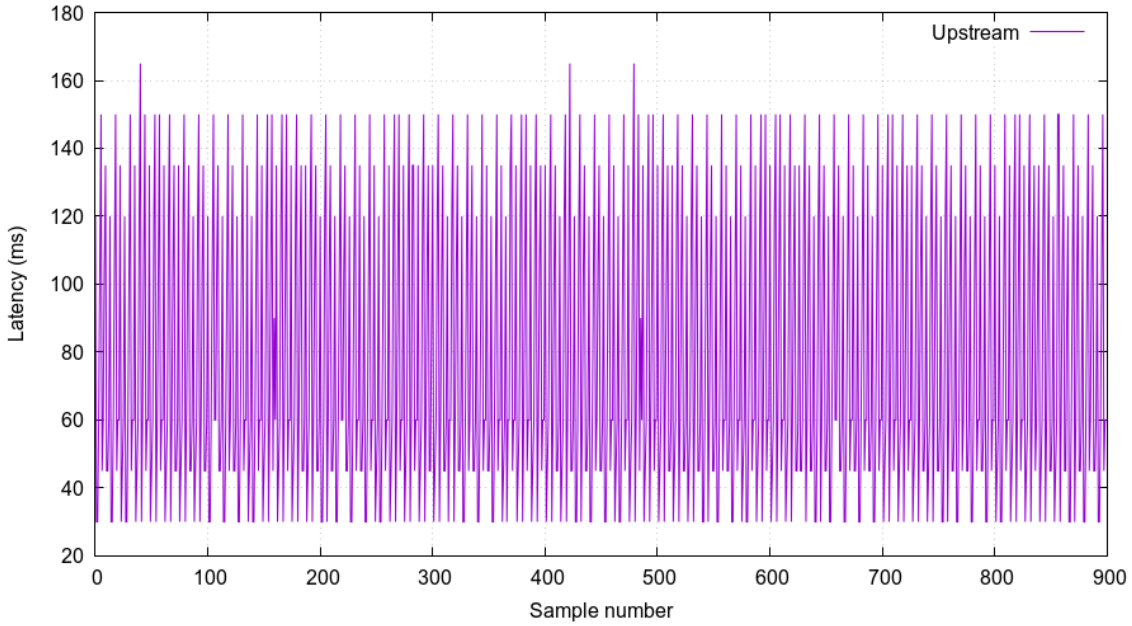


Figure L.4: UWB upstream latency for mote R4 at the UNB ITC building.

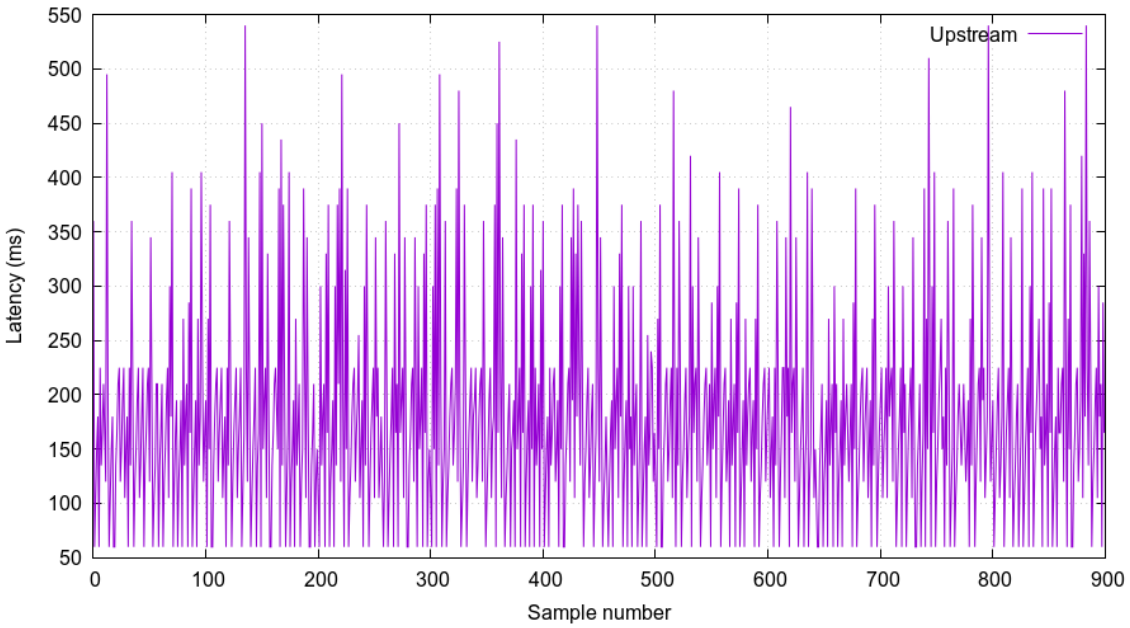


Figure L.5: UWB upstream latency for mote R5 at the UNB ITC building.

L.1.1.2 WirelessHART

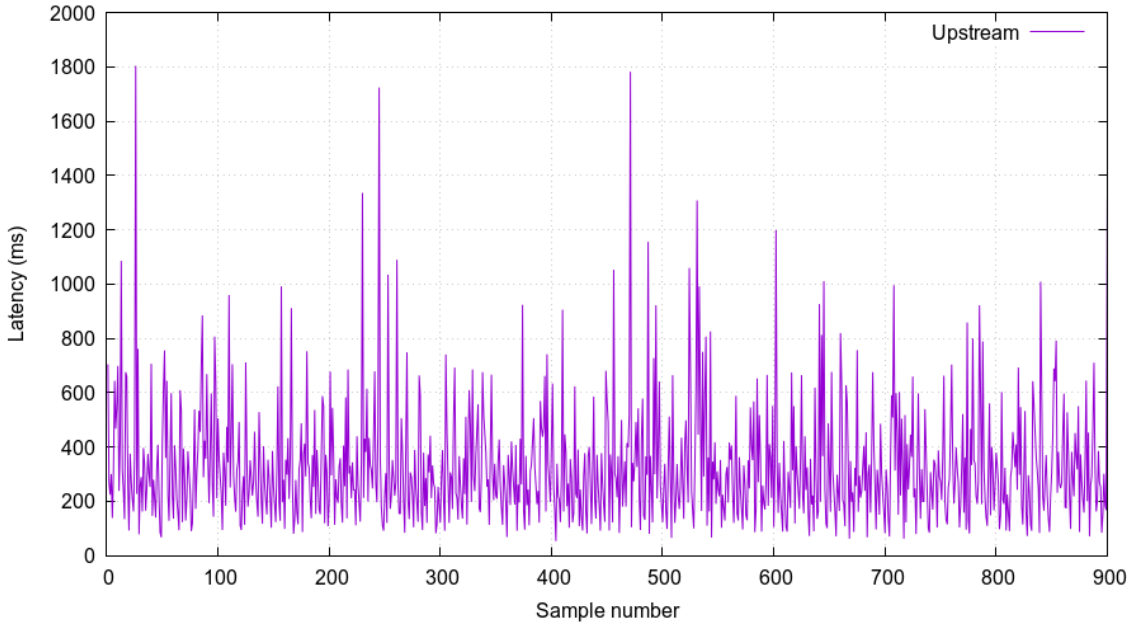


Figure L.6: WirelessHART upstream latency for mote R1 at the UNB ITC building.

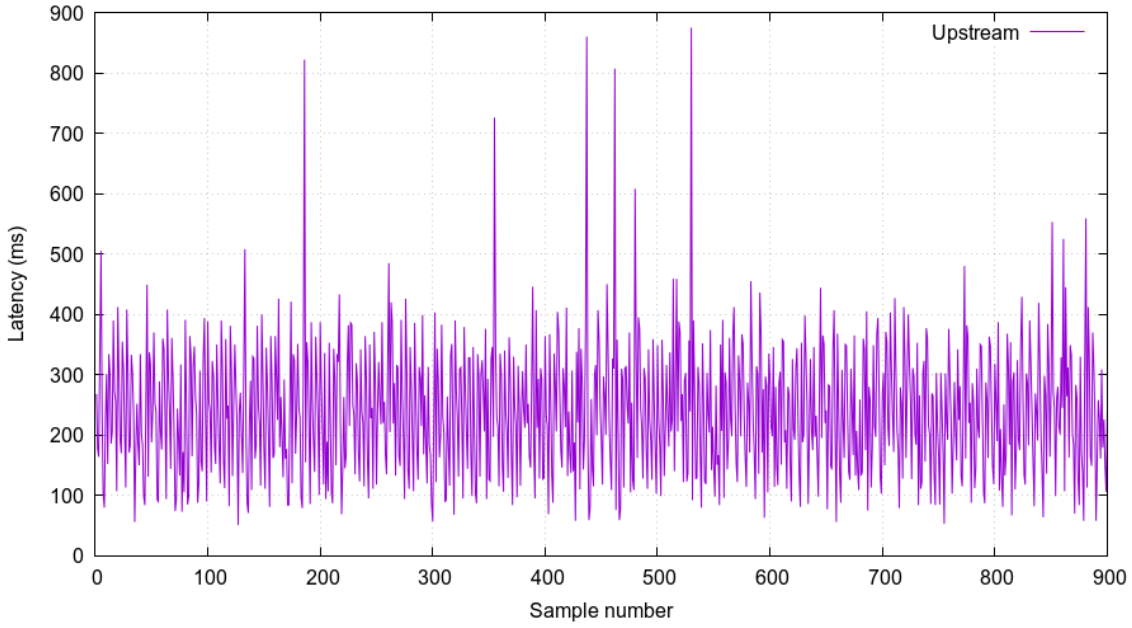


Figure L.7: WirelessHART upstream latency for mote R2 at the UNB ITC building.

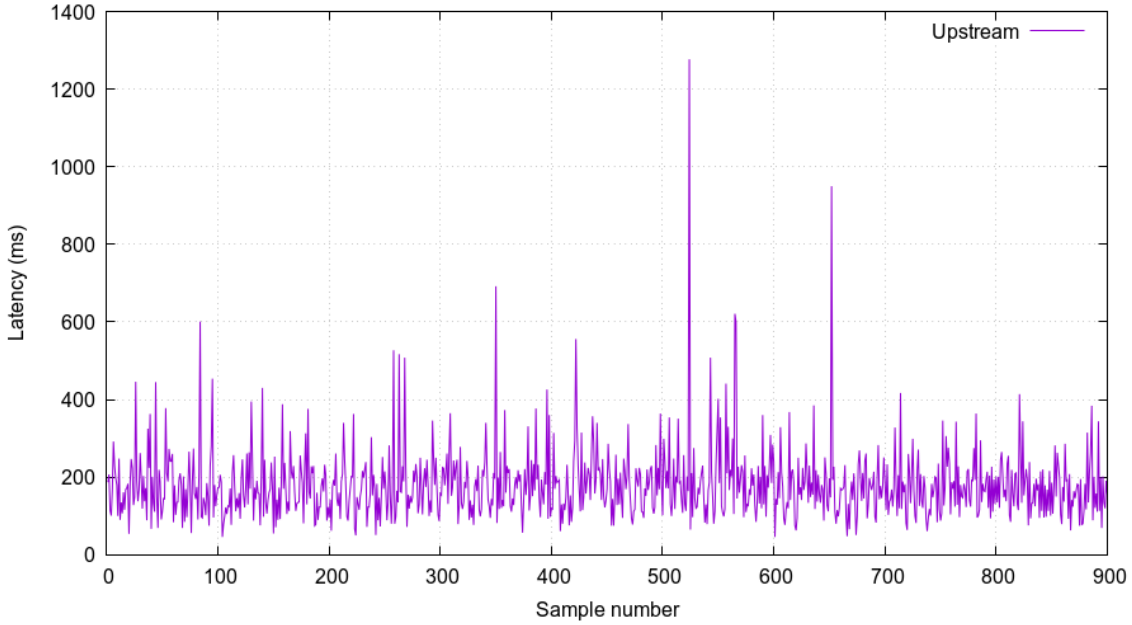


Figure L.8: WirelessHART upstream latency for mote R3 at the UNB ITC building.

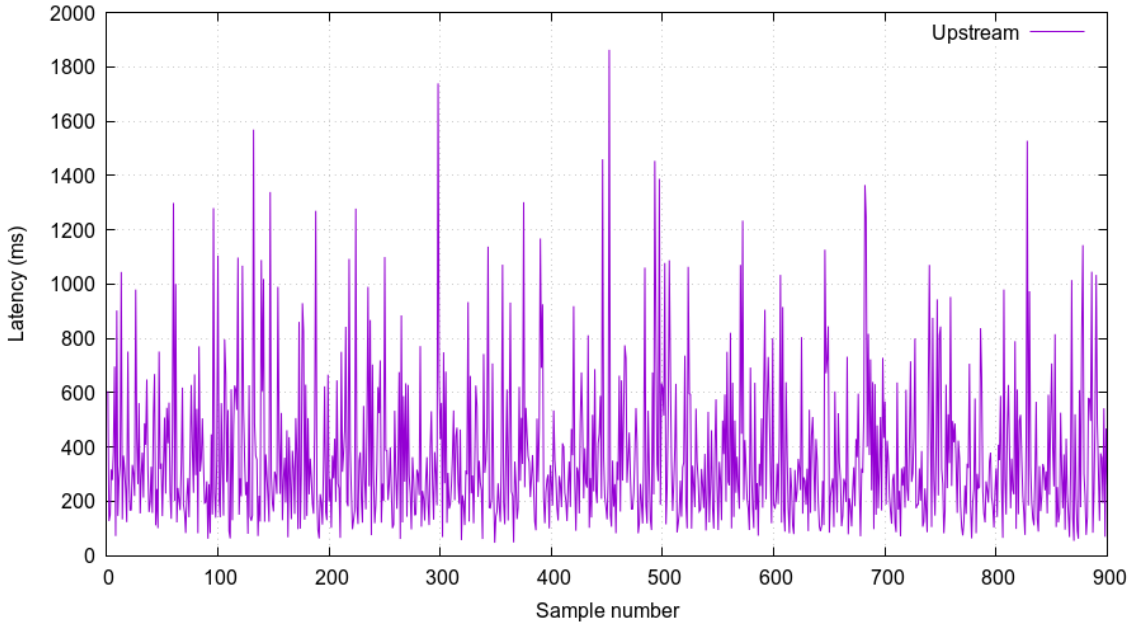


Figure L.9: WirelessHART upstream latency for mote R4 at the UNB ITC building.

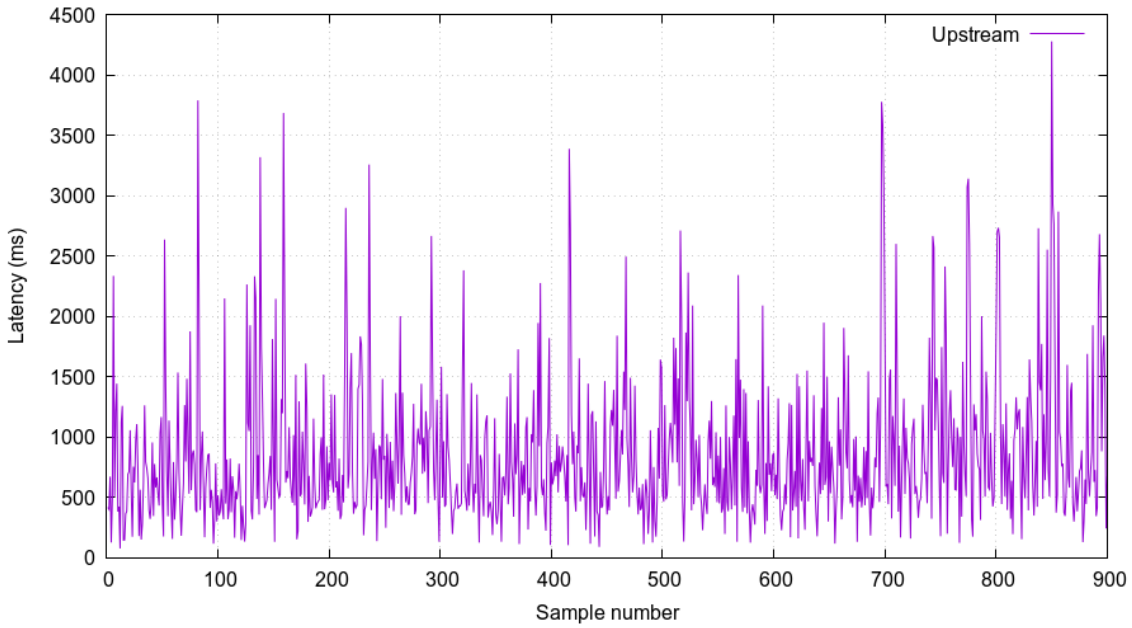


Figure L.10: WirelessHART upstream latency for mote R5 at the UNB ITC building.

L.1.2 UNB Heating Plant

L.1.2.1 UWB

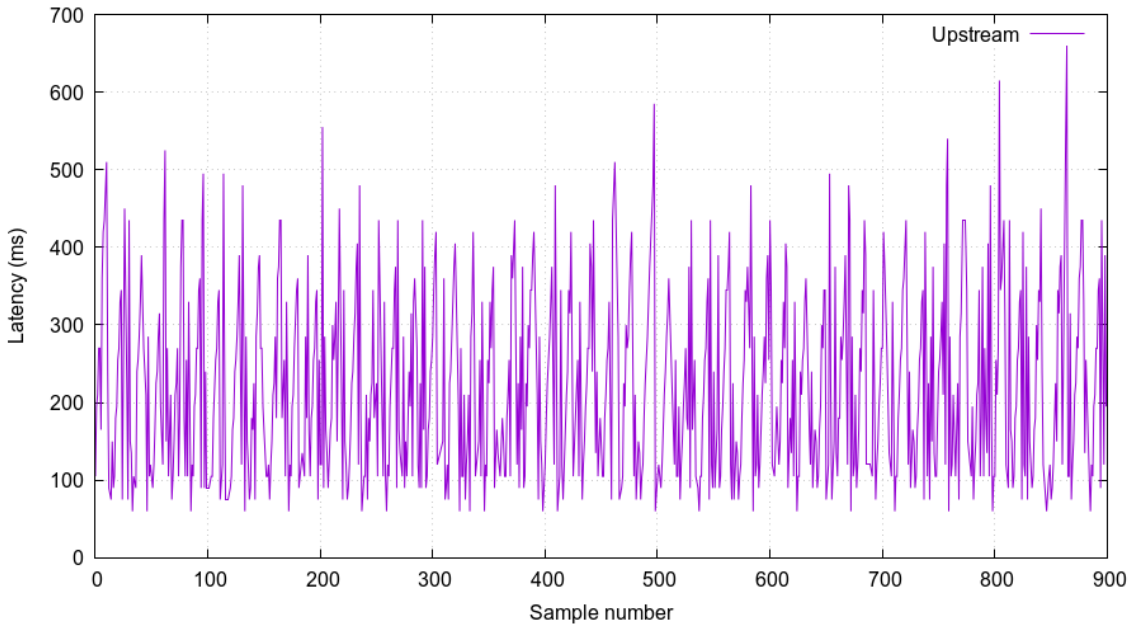


Figure L.11: UWB upstream latency for mote R1 at the UNB heating plant.

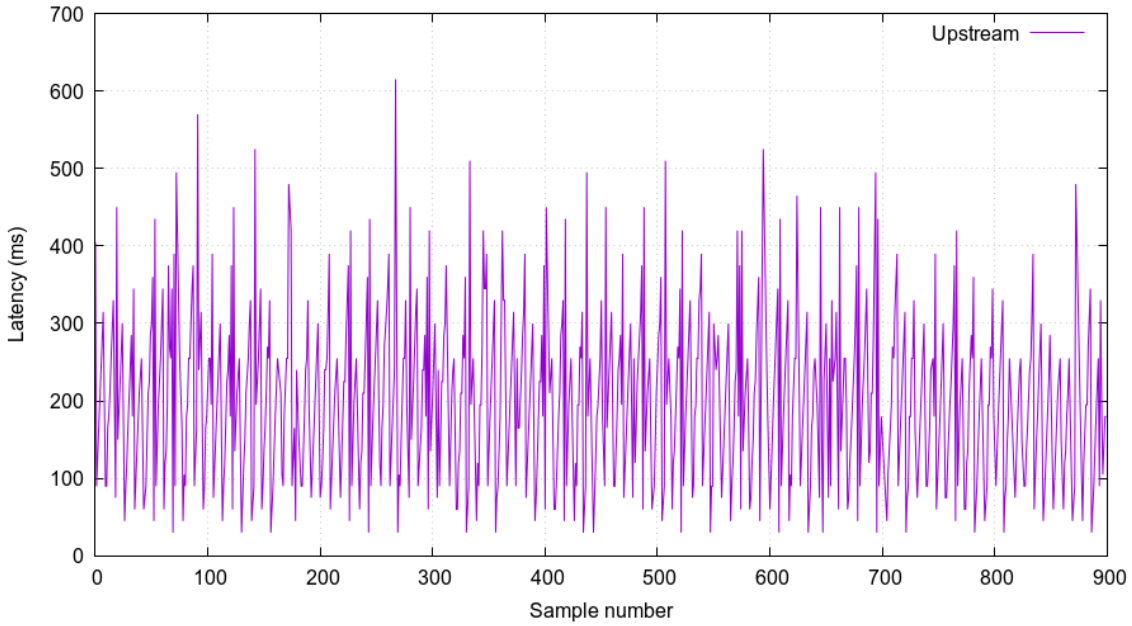


Figure L.12: UWB upstream latency for mote R2 at the UNB heating plant.

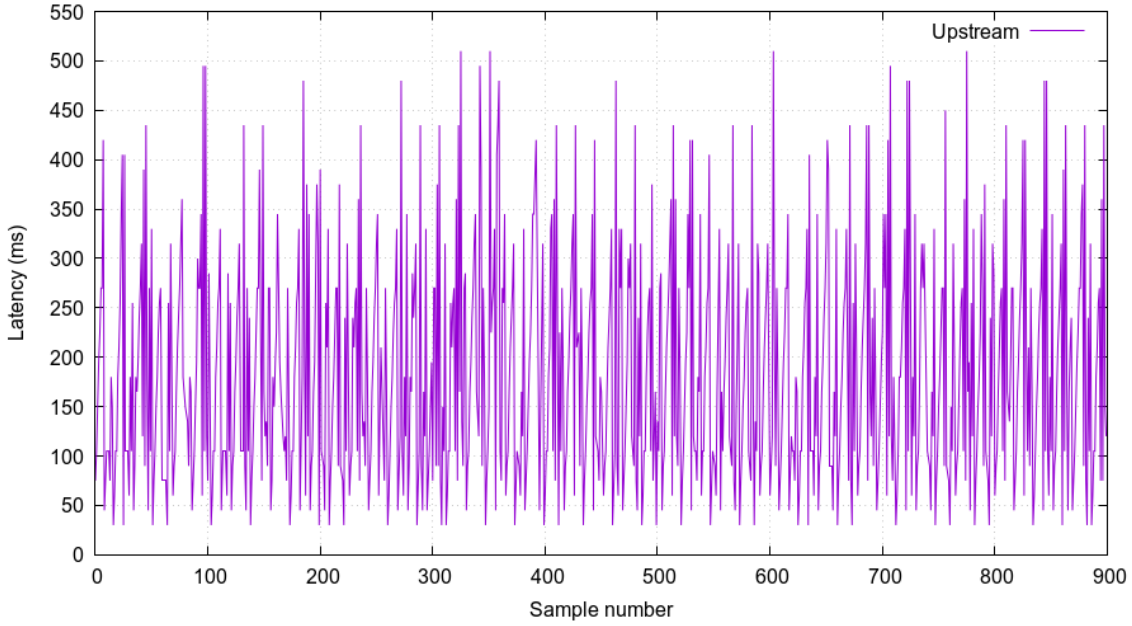


Figure L.13: UWB upstream latency for mote R3 at the UNB heating plant.

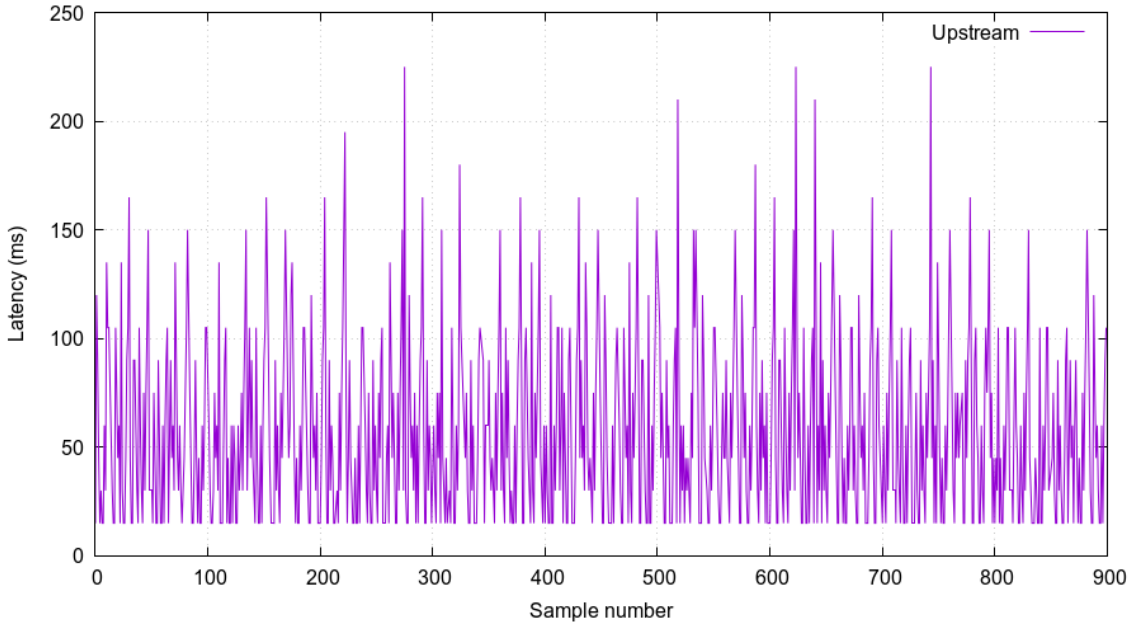


Figure L.14: UWB upstream latency for mote R4 at the UNB heating plant.

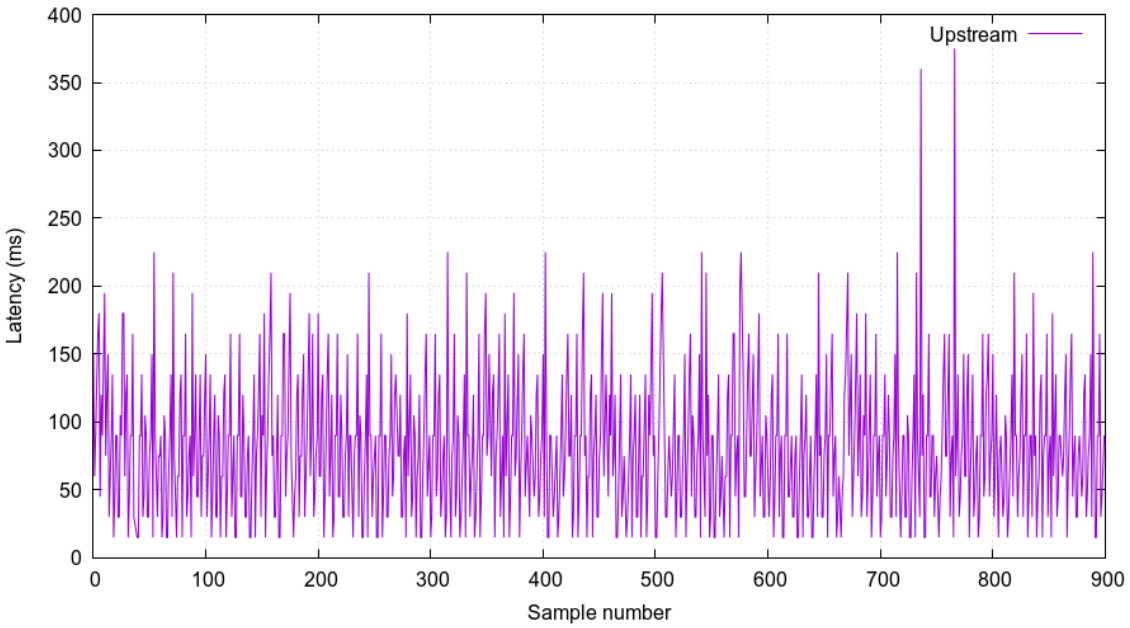


Figure L.15: UWB upstream latency for mote R5 at the UNB heating plant.

L.1.2.2 WirelessHART

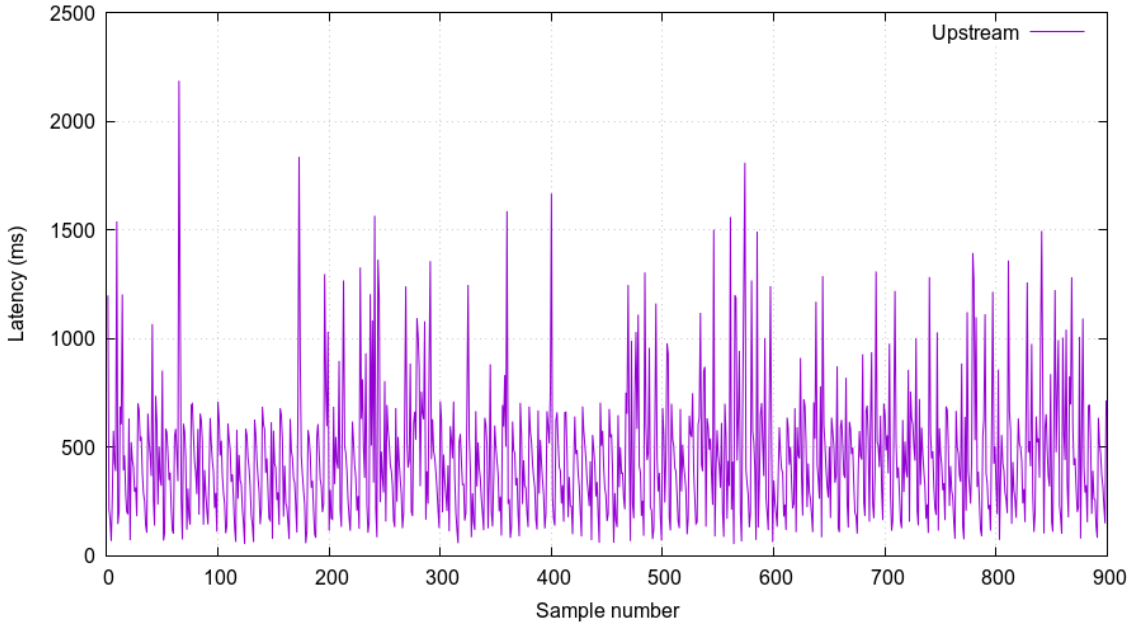


Figure L.16: WirelessHART upstream latency for mote R1 at the UNB heating plant.

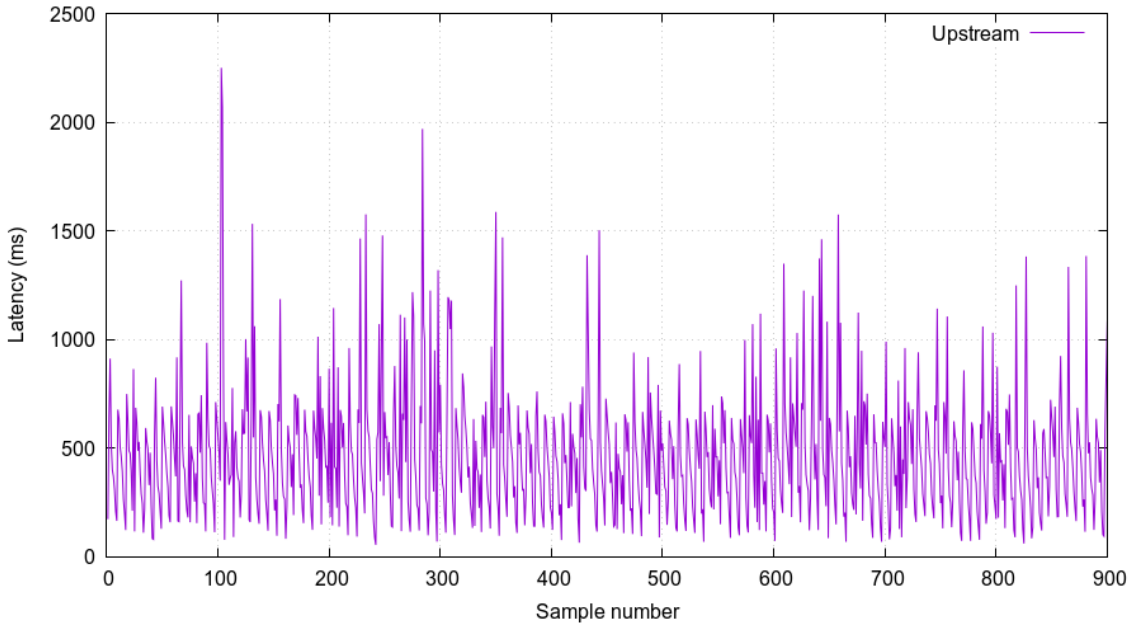


Figure L.17: WirelessHART upstream latency for mote R2 at the UNB heating plant.

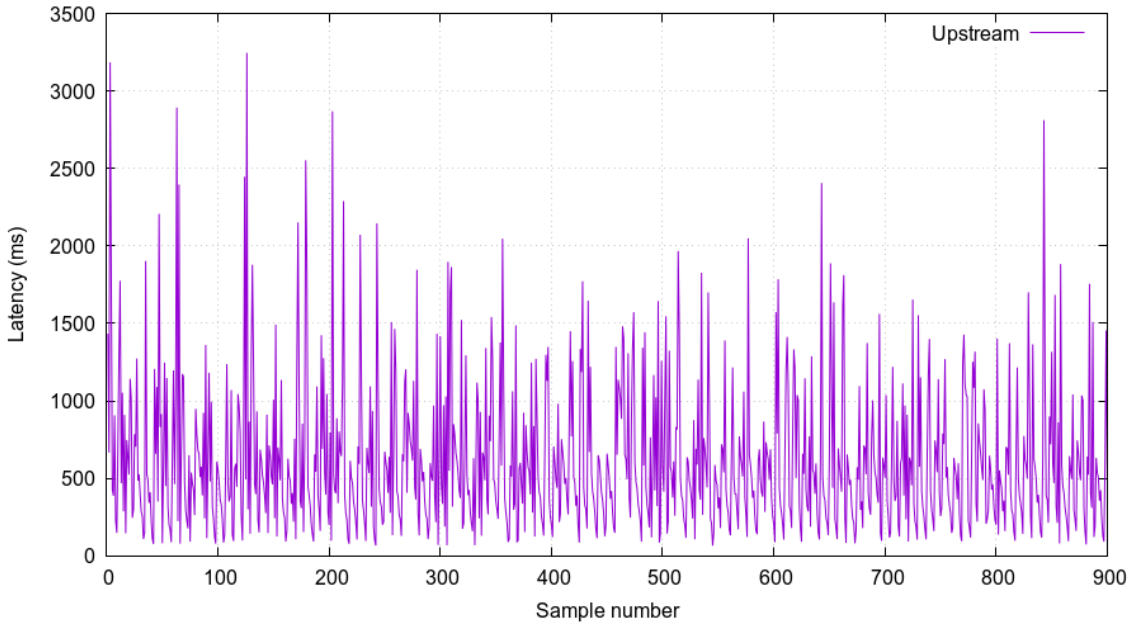


Figure L.18: WirelessHART upstream latency for mote R3 at the UNB heating plant.

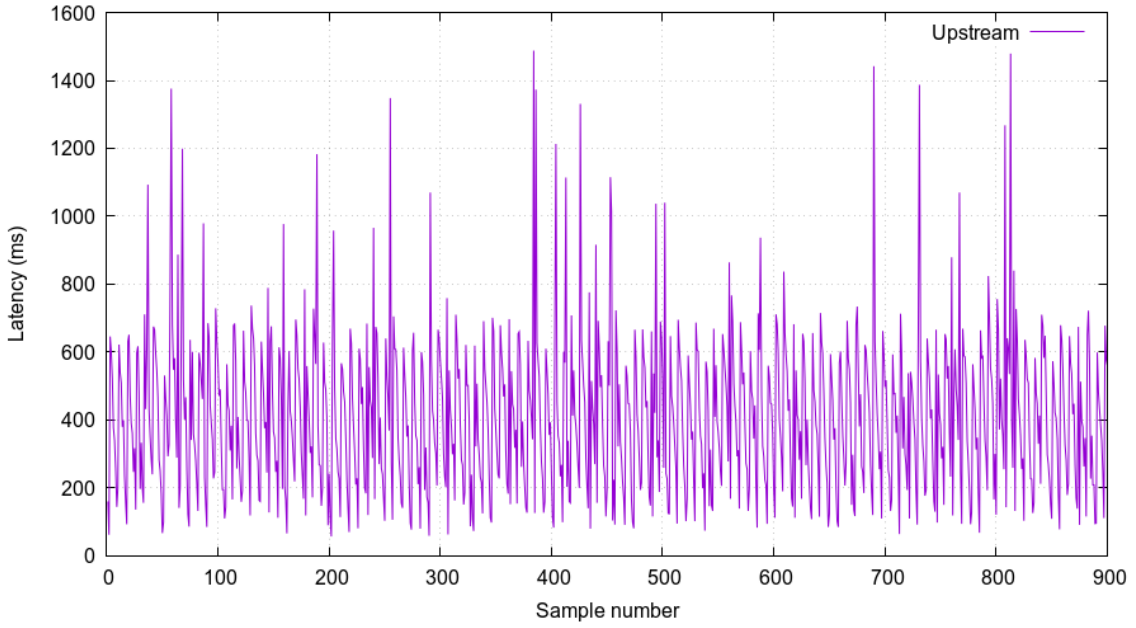


Figure L.19: WirelessHART upstream latency for mote R4 at the UNB heating plant.

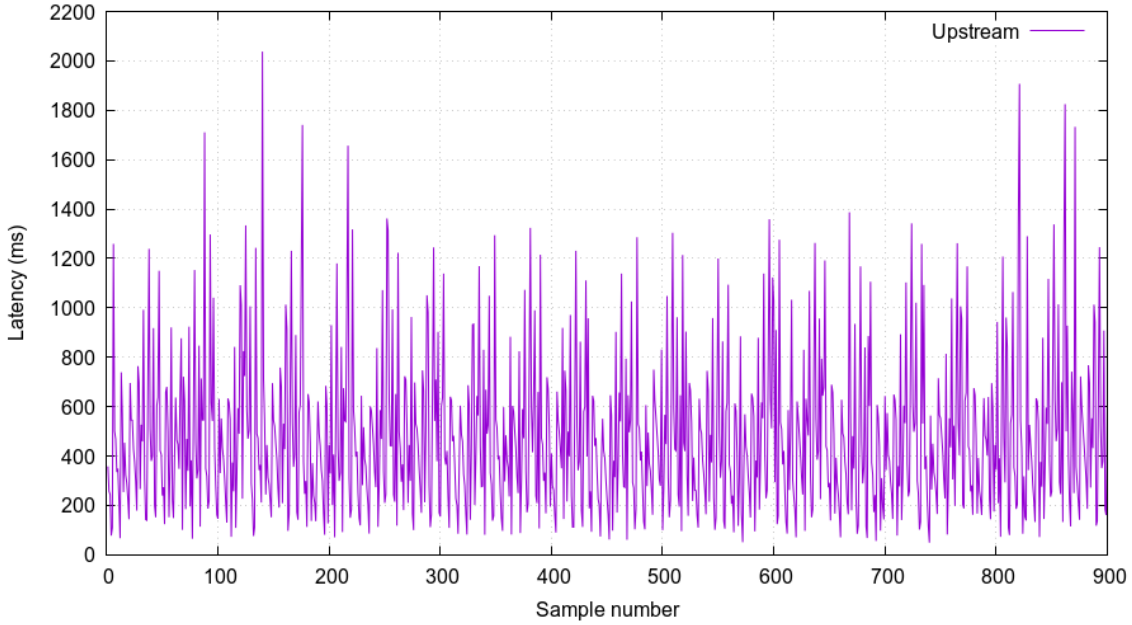


Figure L.20: WirelessHART upstream latency for mote R5 at the UNB heating plant.

L.2 Round Trip Latency

L.2.1 UNB ITC Building

L.2.1.1 UWB Motes

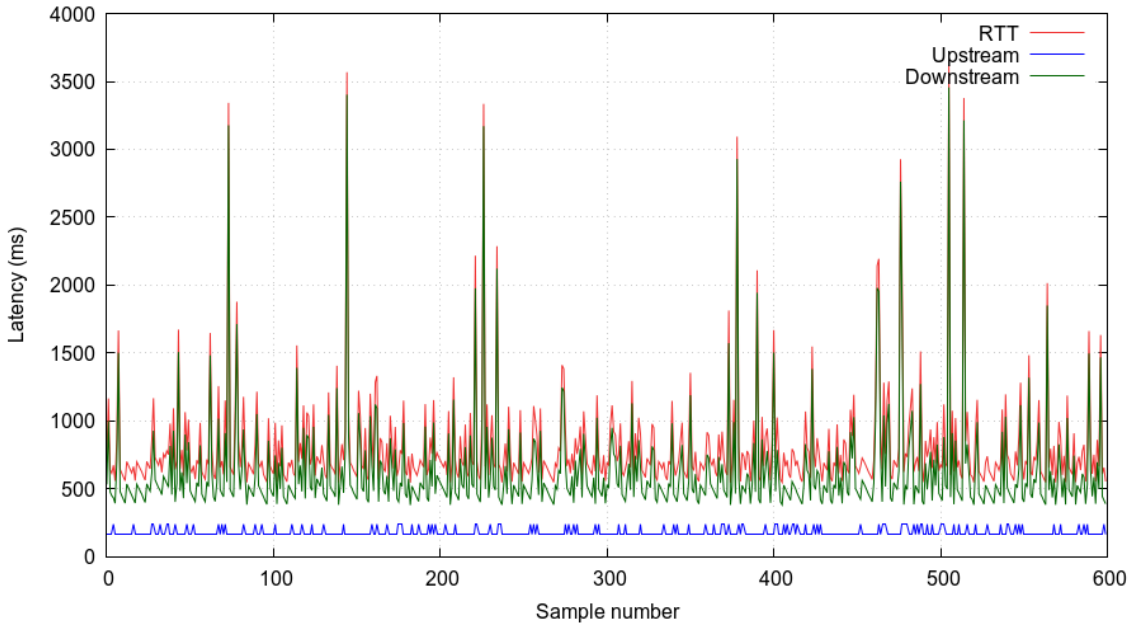


Figure L.21: UWB round trip latency for mote R1 at the UNB ITC building.

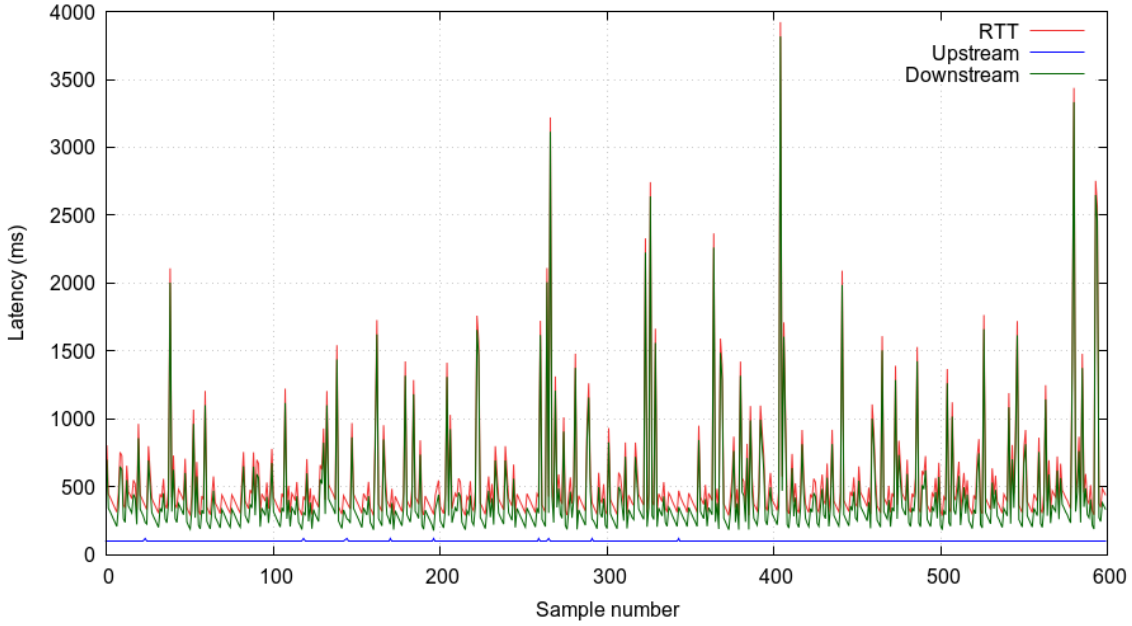


Figure L.22: UWB round trip latency for mote R3 at the UNB ITC building.

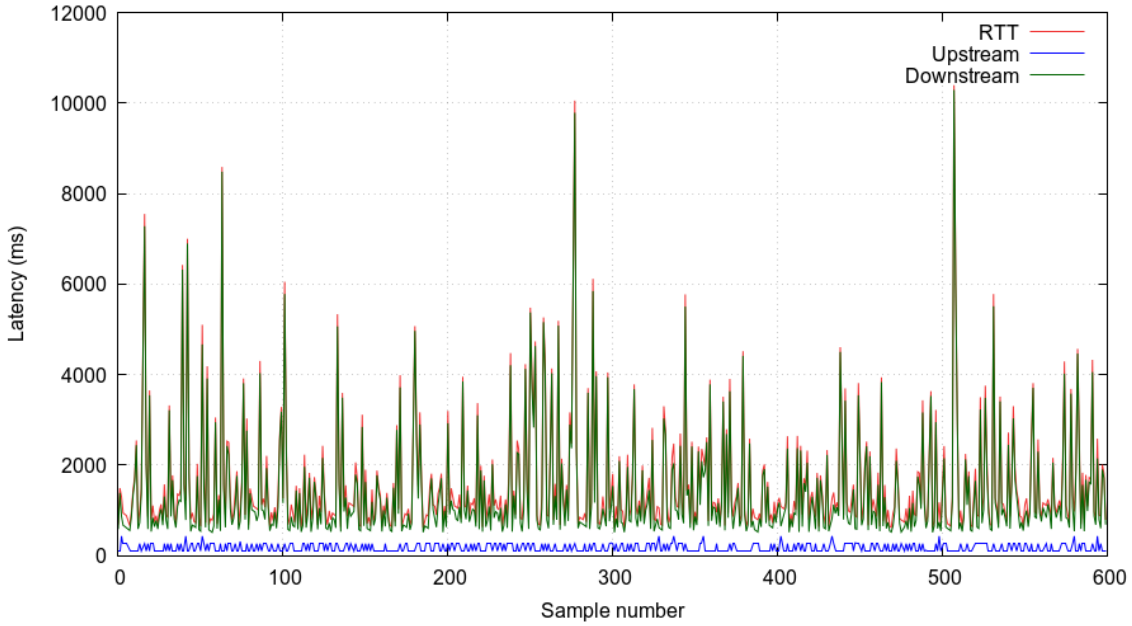


Figure L.23: UWB round trip latency for mote R5 at the UNB ITC building.

L.2.1.2 WirelessHART Motes

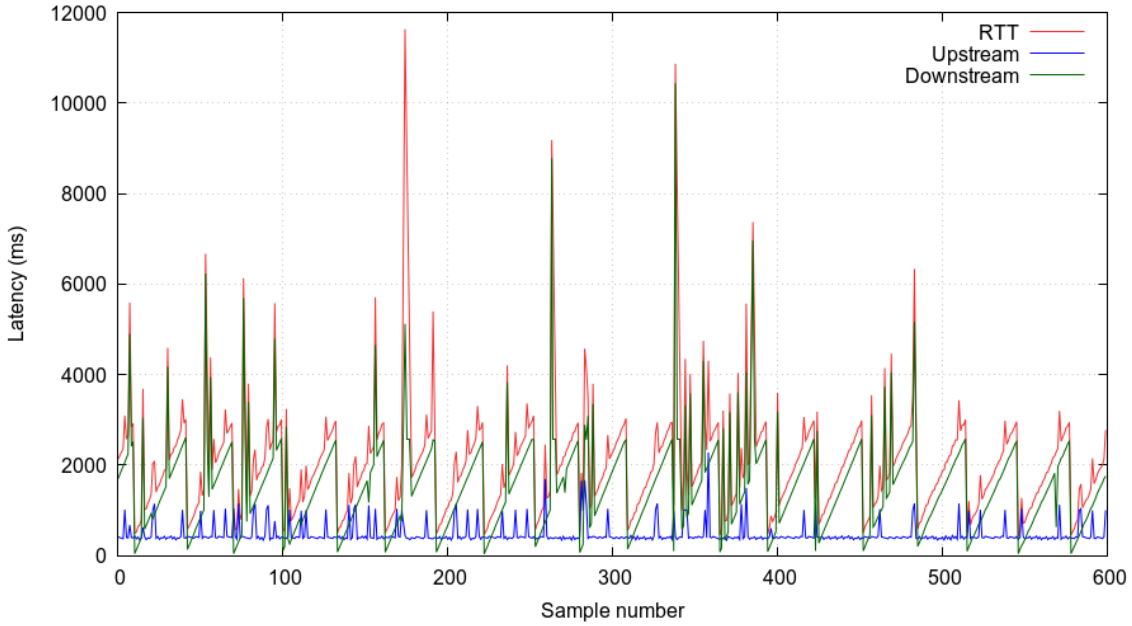


Figure L.24: WirelessHART round trip latency for mote R1 at the UNB ITC building.

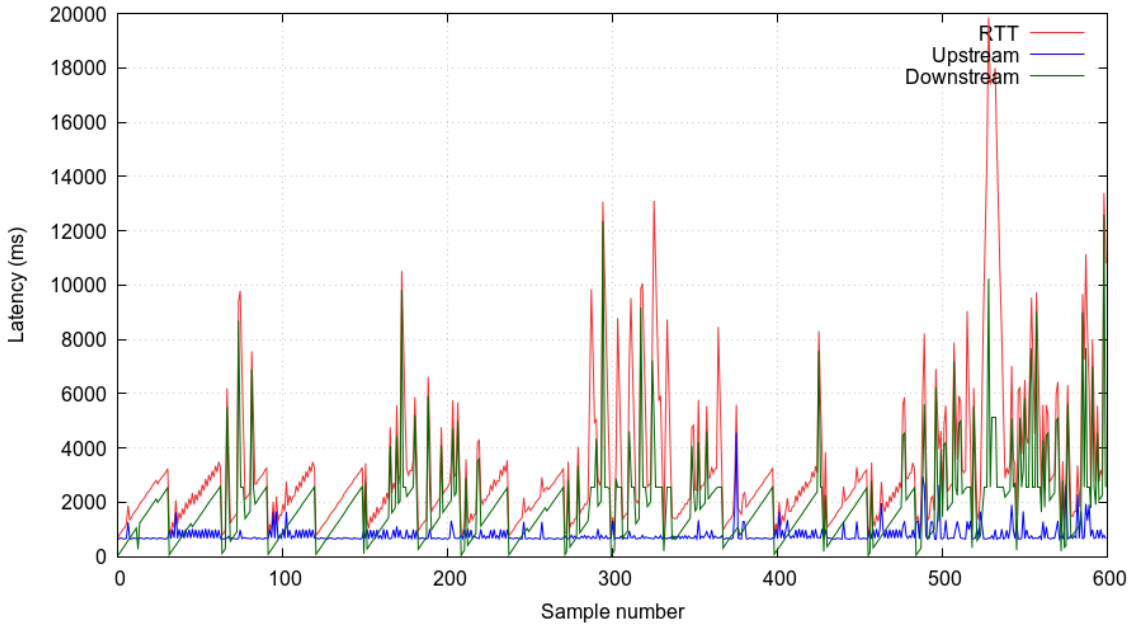


Figure L.25: WirelessHART round trip latency for mote R3 at the UNB ITC building.

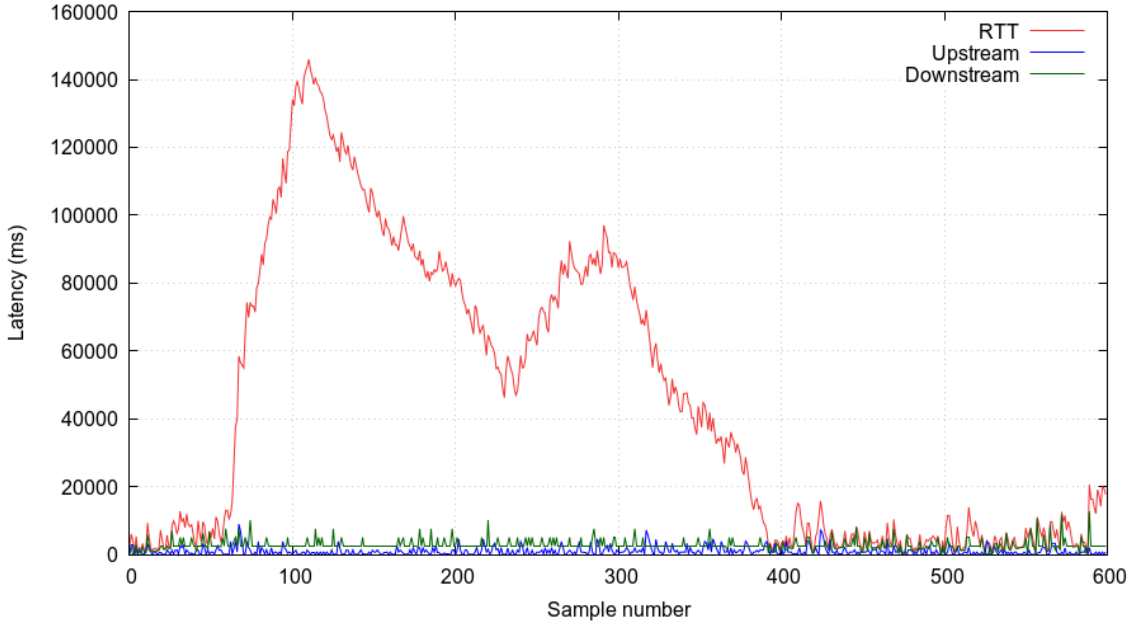


Figure L.26: WirelessHART round trip latency for mote R5 at the UNB ITC building.

L.2.2 UNB Heating Plant

L.2.2.1 UWB Motes

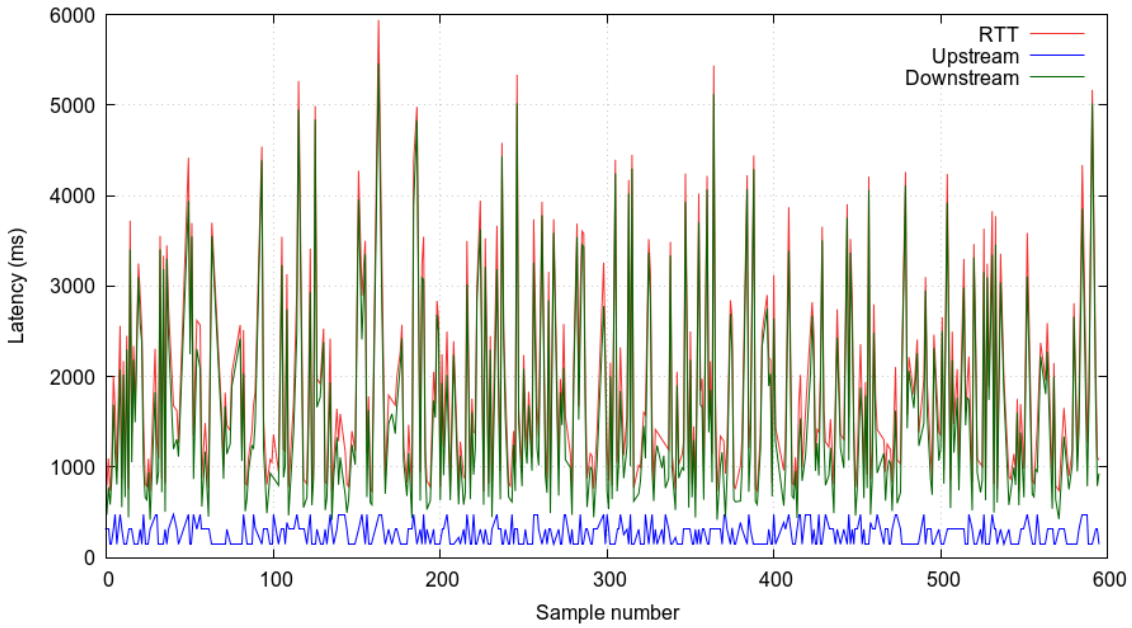


Figure L.27: UWB round trip latency for mote R3 at the UNB heating plant.

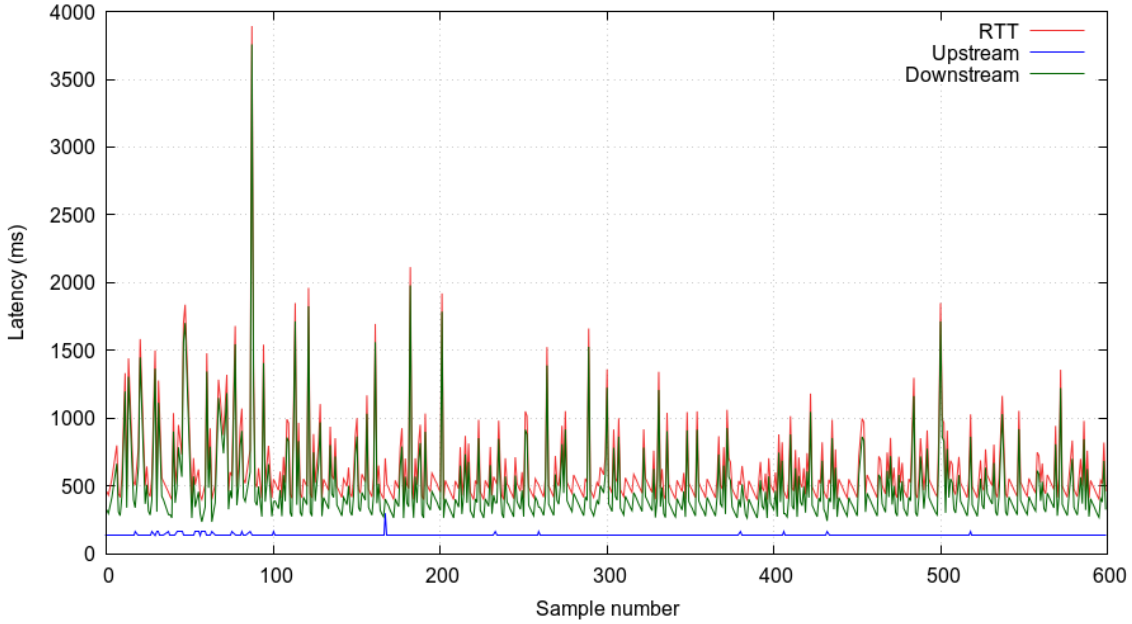


Figure L.28: UWB round trip latency for mote R5 at the UNB heating plant.

L.2.2.2 WirelessHART Motes

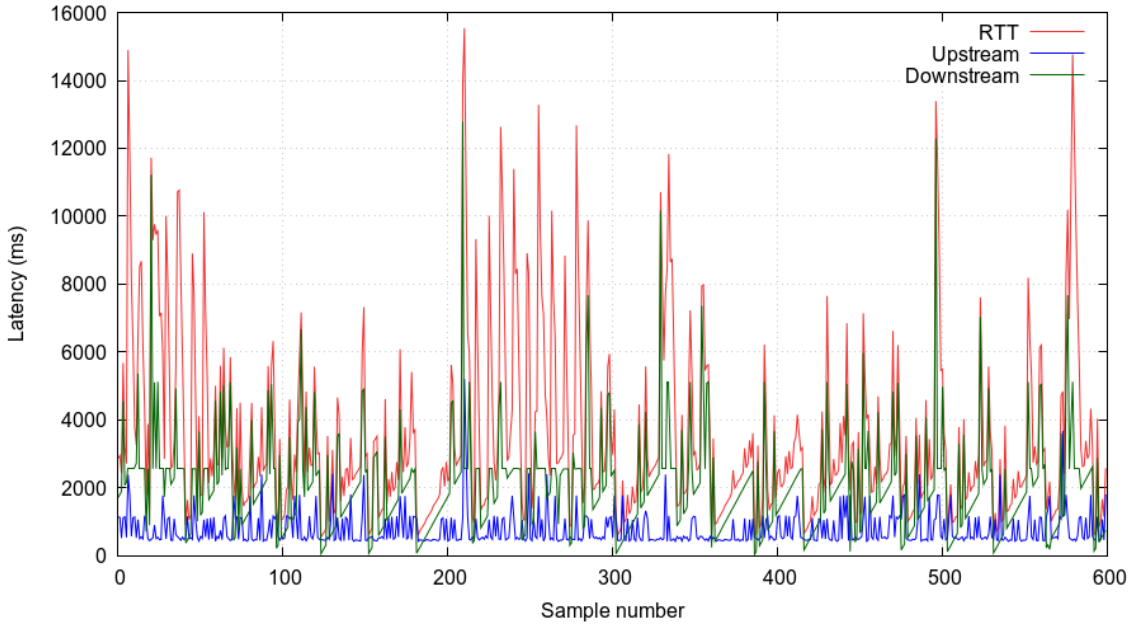


Figure L.29: WirelessHART round trip latency for mote R3 at the UNB heating plant.

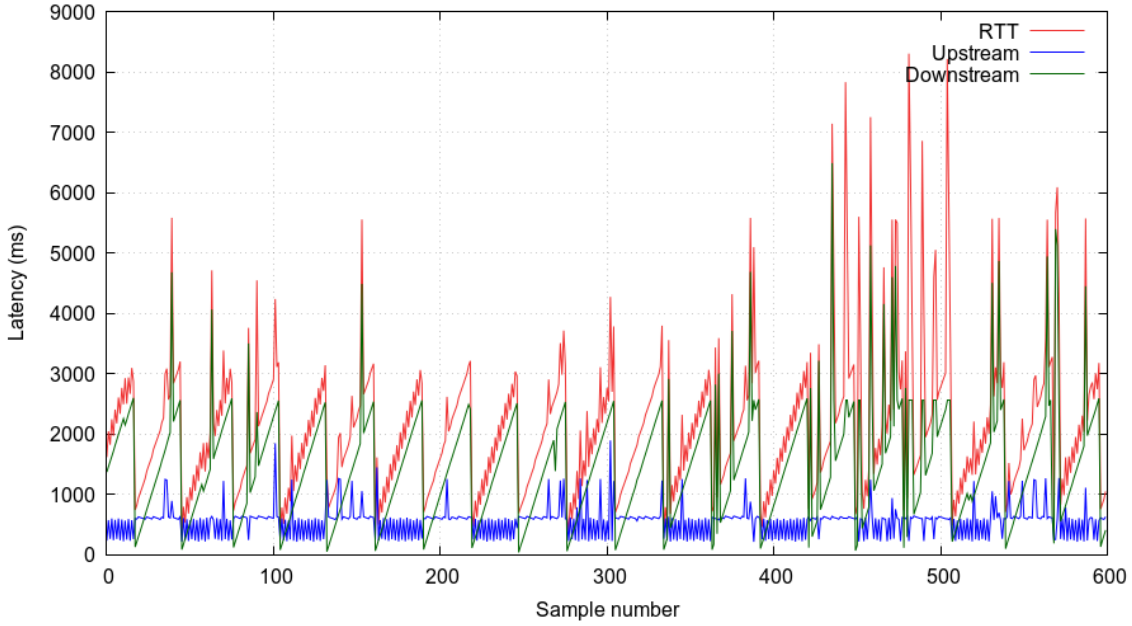


Figure L.30: WirelessHART round trip latency for mote R5 at the UNB heating plant.

Appendix M

Average Latency Measurements

This section presents the average upstream latency μ_u, average round trip time μ_{RTT}, and average downstream latency μ_d for all tests performed at both test locations.

M.1 Upstream Scenario Average Latency

Table M.1: Average upstream test latency in ms at the UNB ITC building.

	Mote #1		Mote #2		Mote #3		Mote #4		Mote #5	
	UWB	WH	UWB	WH	UWB	WH	UWB	WH	UWB	WH
μ_u	163.48	412.34	52.68	301.94	42.58	181.11	123.30	216.29	289.56	459.79
σ_u	70.25	288.58	37.78	285.77	28.93	218.32	47.40	145.23	119.20	225.92
Min.	75	69	15	46	15	49	45	51	120	95
Max.	510	4594	165	4832	150	5133	270	2187	675	2187

Table M.2: Upstream test latency in ms at the UNB heating plant.

	Mote #1		Mote #2		Mote #3		Mote #4		Mote #5	
	UWB	WH	UWB	WH	UWB	WH	UWB	WH	UWB	WH
μ_u	219.97	455.45	204.35	467.81	185.56	615.25	56.15	410.84	81.99	479.83
σ_u	117.03	305.12	112.28	294.34	115.04	482.28	40.77	230.25	51.79	319.95
Min.	60	54	30	55	30	66	15	57	15	49
Max.	660	2186	615	2250	510	3246	225	1488	375	2038

M.2 Round Trip Scenario Average Latency

Table M.3: Average RTT in ms at the UNB ITC building.

	Mote #1		Mote #3		Mote #5	
	UWB	WH	UWB	WH	UWB	WH
μ_{RTT}	810.45	2087.91	541.13	3242.79	1574.60	44418.65
σ_{RTT}	381.77	1276.70	411.55	2653.24	1260.41	42157.63
Min.	546	425	289	722	622	211
Max.	3621	11631	3922	19858	10395	145936

Table M.4: RTT test average upstream latency in ms at the UNB ITC building.

	Mote #1		Mote #3		Mote #5	
	UWB	WH	UWB	WH	UWB	WH
μ_u	178.94	475.39	105.23	810.56	175.65	1106.05
σ_u	29.20	229.01	1.84	318.00	87.89	1233.23
Min.	165	344	105	645	105	77
Max.	240	2282	120	4571	435	9041

Table M.5: RTT test estimated average downstream latency in ms at the UNB ITC building.

	Mote #1		Mote #3		Mote #5	
	UWB	WH	UWB	WH	UWB	WH
μ_d	631.50	1531	435.90	1924.30	1398.95	2869.39
σ_d	381.94	1070.66	411.64	1616.85	1256.14	1574.63
Min.	381	41	178	42	517	92
Max.	3456	10435	3817	12596	10290	12805

Table M.6: Average RTT in ms at the UNB heating plant.

	Mote #3		Mote #5	
	UWB	WH	UWB	WH
μ_{RTT}	1853.11	3556.01	616.87	2137.77
σ_{RTT}	1080.74	2681.13	304.04	1177.76
Min.	730	596	399	319
Max.	5939	15540	3893	8302

Table M.7: RTT test average upstream latency in ms at the UNB heating plant.

	Mote #3		Mote #5	
	UWB	WH	UWB	WH
μ_u	261.76	722.61	136.53	553.97
σ_u	121.18	490.35	9.13	240.30
Min.	150	425	135	215
Max.	480	5205	300	1895

Table M.8: RTT test estimated average downstream latency in ms at the UNB heating plant.

	Mote #3		Mote #5	
	UWB	WH	UWB	WH
μ_d	1591.35	2075.18	480.34	1488.7
σ_d	1077.38	1527.18	304.04	937.86
Min.	418	39	236	40
Max.	5459	12795	3758	6489

Vita

Candidate's full name:
Daniel Mark King

University attended:
Master of Computer Science (2015 – 2017)
University of New Brunswick
Fredericton, Canada

BSc (Hons) Computer Science (2006 – 2009)
University of Essex
Colchester, United Kingdom

Conference Poster Presentations:
Daniel M. King and Bradford G. Nickerson and Wei Song. Real-time Wireless Control via Ultra-Wideband (UWB) Communication. University of New Brunswick Computer Science Research Expo 2017, Fredericton, Canada

Daniel M. King and Bradford G. Nickerson. Real-time Wireless Control via Ultra-Wideband (UWB) Communication. University of New Brunswick Computer Science Research Expo 2016, Fredericton, Canada

Other Documents:
Daniel M. King and Bradford G. Nickerson, Ultra-Wideband Wireless Communication for Real-Time Control, Technical Report TR16-238, University of New Brunswick, Faculty of Computer Science, May 2016.