

https://dspace.org/

Wireless mesh network data communications and reliability analysis for application deployment in anti-theft solutions

Van Rensburg, Roger

2020
https://hdl.handle.net/10539/32691

Downloaded from a DSpace repository. https://dspace.org/

Wireless mesh network data communications and reliability analysis for application deployment in anti-theft solutions

by

Roger van Rensburg



WITS
UNIVERSITY

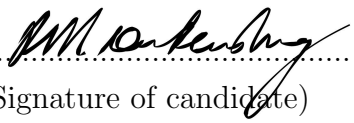
Submitted to the Faculty of Science
in partial fulfilment of the requirements
for the degree of Master of Science

Supervisor: Prof. Bruce Mellado

September 7, 2020

Declaration

"I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Science at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University."



(Signature of candidate)

Submitted: 7 September 2020

This dissertation is dedicated
to my best friend and soulmate,
Lehana

Abstract

This research is an initiative from the High-Throughput Electronics Laboratory at the University of the Witwatersrand to develop low-power wireless mesh networks for reliable data communications in the design of innovative anti-theft solutions. A network test-bench was developed, where millions of packets were transmitted and logged between interconnected nodes to analyse the quality of the network's service in a harsh indoor building environment. Similar methodologies as found in 'big data' analysis in particle physics were adopted to analyse the network's performance and reliability. The quality of service between multiple asynchronous transmitting nodes was determined and compared with the wireless technology routing protocol to assess coverage in large geographical areas. The statistical analysis conducted for each route and distributions revealed one, two and three-hop packet transmissions have approximately 10, 20 and 30 ms round-trip times, respectively. Multimodal distributions in round-trip times of routes were present, consisting of an overlay of multiple normal distributions that indicate the presence of dynamic routing. The higher percentage of round-trip times for routes that fall outside three standard deviations from the mean of the distributions indicated reliability issues and was found more susceptible to packet loss. Overall, the wireless technology provided a total packet delivery ratio of 98.2% and found compatible with an IEEE 802.15.4 based low-power mesh networking protocol for low-throughput wireless communications. The deployment of anti-theft solutions is feasible in indoor or outdoor environments and the network can be used for various instrumentation applications in the control and monitoring of intelligent embedded systems.

Acknowledgements

The anti-theft initiative is made possible by the High-Throughput Electronics Laboratory at the School of Physics. I acknowledge my supervisor, Prof. Bruce Mellado and project advisor, Dr Cesar Augusto Tobon, who initially brainstormed an anti-theft solution and involving me in this innovative project undertaking.

My sincerest gratitude is extended to Wits Enterprise in their active involvement during the inception phase of this project. A special thanks to the Faculty of Science and the University of Witwatersrand for their financial assistance.

This research would not have been possible without SA-CERN funded by the Department of Science and Technology and the National Research Foundation. Finally, I would like to acknowledge students, Rorisang Sitoboli, Edward Nkadimeng and Lebohang Mokoena who participated in the data analysis collected in this study.

Research Outputs

The following research outputs were produced during the period of the study:

i.) High Energy Particle Physics Workshop
1–3 February 2017, Johannesburg, South Africa
Oral presentation and paper - Published: 8 September 2017
Available online at:
http://iopscience.iop.org/article/10.1088/1742-6596/889/1/012016

ii.) The Proceedings of SAIP2017, the 62nd Annual Conference of the South African Institute of Physics, Applied Physics
3–7 July 2017, Stellenbosch, South Africa
Poster presentation and paper - Published: 31 December 2018
Available online at:
http://events.saip.org.za/getFile.py/access?resId=64&materialId=8&confId=91

iii.) The Proceedings of SAIP2018, the 63rd Annual Conference of the South African Institute of Physics, Applied Physics
25-29 June 2018, Bloemfontein, South Africa
Oral presentation only

iv.) Journal of Engineering, Design and Technology
Paper - Published 28 January 2019
Available online at:
https://www.emeraldinsight.com/doi/full/10.1108/JEDT-08-2018-0138

Table of Contents

DECLARATION ii

ABSTRACT iv

ACKNOWLEDGEMENTS v

RESEARCH OUTPUTS vi

LIST OF FIGURES xiv

LIST OF TABLES xv

LIST OF EQUATIONS xvi

Acronyms and Abbreviations xvii

Chapter 1: Introduction 1

1.1 LHC at CERN 2

1.1.1 LHC superconductor magnets 3

1.1.2 LHC protection system 5

1.1.3 Concept of IoT at the LHC 7

1.1.4 Conclusion 8

1.2 The HTEL anti-theft initiative 9

1.2.1 PCD anti-theft solutions 10

1.2.2 Proposed IoT at the HTEL 14

1.2.3 Conclusion . . . 16
1.3 Problem Statement . . . 18
1.3.1 Low-power wireless technologies . . . 18
1.3.2 Mesh network development . . . 18
1.3.3 Mesh network characterisation . . . 19
1.4 Hypothesis . . . 19
1.5 Methodology . . . 20
1.6 Importance of Research . . . 21
1.7 Research Delineation of the ICPP . . . 22
1.8 Outline of Dissertation . . . 23

Chapter 2: Low-Power Wireless Technologies 24
2.1 Concept-to-implementation phase . . . 25
2.2 Low-Power Wireless Technologies . . . 31
2.3 Bluetooth low-energy . . . 37
2.3.1 IEEE 802.15.1 . . . 42
2.3.2 Protocol stack . . . 42
2.3.3 Network topology . . . 52
2.4 OpenThread . . . 56
2.4.1 IEEE 802.15.4 . . . 57
2.4.2 Protocol stack . . . 60
2.4.3 Network topology . . . 66
2.5 IEEE 802.15.4 Proprietary . . . 69
2.6 Conclusion . . . 69

Chapter 3: Mesh Network Development 71
3.1 Methodology . . . 71
3.2 Experimental Study . . . 72
3.2.1 Embedded system and RF design . . . 73
3.2.1.1 Hardware development . . . 73
3.2.1.2 Software development . . . 76

3.2.1.3 Power consumption 78
3.2.2 Router distribution 79
3.2.3 Mesh network development 81
3.3 Conclusion 86

Chapter 4: Results **87**
4.1 Data Collection 87
4.1.1 Data analysis 88
4.1.2 Performance and reliability metrics 89
4.1.3 Performance analysis 91
4.1.4 Reliability analysis 105
4.1.5 Low-throughput versus RTT 106
4.1.6 Routing algorithm simulation 107
4.2 Conclusion 112

Chapter 5: Discussion and Conclusion **113**
5.1 Statement of Objectives 114
5.2 Conclusion 114
5.3 Discussion 117
5.4 Future Work 118

References **120**

Appendices **128**
Appendix I: PCD anti-theft solutions. 129
Appendix II: nRF52840 PCB design guidelines. 131
Appendix III: ICMP algorithm and logging C code. 144
Appendix IV: Gaussian kernel Matlab script. 159

List of Figures

1.1 CERN's accelerator complex (Image by CERN). 2
1.2 LHC dipole depict in a) a three-dimensional view of the superconductive cable surrounding two proton beam pipes travelling in opposite directions and b) the cross-section of the 15-meter cryodipole with component descriptions (Images by CERN). 4
1.3 Proposed HTEL anti-theft solution where tablets connect to the nearest node based on their location in multiple mesh networks. 12
1.4 Block diagram illustrating hardware and software systems for data communications from an Android device to the nearest node in an IEEE 802.15.4 based mesh network. 13
1.5 Block diagram of proposed HTEL anti-theft solution illustrating main technology systems. 15
2.1 Preliminary design concept using an RSIC mounted on the PCD and communicating via BLE to a central node and tablet [20, p.2]. An energy harvester denoted by EH, provides charging capabilities to a battery cell that provides power to the slave nodes. 25
2.2 Final design concept omitting the RSIC mounted on the PCD and communicating via BLE to the nearest node in a mesh network. 30
2.3 Protocol stack of TCP/IP communication system based on the OSI model illustrating the IEEE 802 hardware layer. 31

2.4 TCP/IP protocol suite adapted from Forouzan [29, p.36, 41] with the illustration of multi-hopping packet transmission over a two-hop distance using an intermediary router over two links. A packet is encapsulated from the source node and decapsulated at the receiving node. 34
2.5 BLE protocol stack adapted from Townsend et al. [30, p.16]. 41
2.6 Conceptual heterodyne a) modulation of binary data s[k] to generate r(t) and b) demodulation of analog signal r(t) to generate s[k]. 43
2.7 Closed-loop fractional-N phase PLL frequency synthesiser control system with $\Delta\Sigma$ modulation modified from Lacaita et al. [33, p.14-72] and Su et al. [34, p.881]. 44
2.8 Revised BLE network topology proposed for the HTEL anti-theft solution illustrating the connection of multiple slaves nodes (PCD-S1 and PCD-S2) representing tablets to a single master node. Master nodes are configured as routers in an IEEE 802.15.4 [19] based mesh topology. 53
2.9 Proposed device shutdown algorithm running on device modified from the preliminary study van Rensburg [20, p.6, Figure 2a]. 54
2.10 Thread protocol stack built on the IEEE 802.15.4 [19] hardware layer. . . . 57
2.11 Conceptual O-QPSK modulation modified from Xiong [32, p.180]. 62
2.12 Conceptual O-QPSK demodulation modified from Xiong [32, p.181]. 63
2.13 Thread network topology proposed for the HTEL anti-theft solution illustrating how multiple IEEE 802.15.4 [19] based mesh networks are connected through redundant gateways over the internet to a cloud service. The Thread end-device nodes are excluded from the network design. 67
3.1 Nordic Semiconductor's a) aQFN73 and b) WLCSP nRF52840-SoC capable of running advanced multi-protocol Bluetooth 5, Thread and ZigBee wireless technologies on a single chip [45]. 72

3.2 nRF52840-DK [48] utilised in the experimental study for fast prototyping and acting as a router in the Thread network. The nRF52840-SoC (labelled U1) with the passive components is mounted on the top layer located inside the square silkscreen marking. The rest of the nRF52840-DK PCB embeds an MCU (labelled U2) for programming and accessing IO ports of the nRF52840-SoC. 74
3.3 Physics building ground floor router distribution determined from log-distance modelling. Each router is mounted on an enclosure to ensure radiation of the omnidirectional RF-modulated signal. 77
3.4 Log-distance modelling of RSSI versus $\log_{10}(\mathrm{d})$ at $\mathrm{T_x} = 0$ dBm. 80
3.5 Mesh network test-bench development at Physics building. 82
3.6 Router $\mathrm{R_s} \rightarrow \mathrm{R_d}$ packet request and logging algorithm. 84
4.1 Router $\mathrm{R_1} \rightarrow \mathrm{R_d}$ RTT distributions a) within 3σ b) outside 3σ and c) $\mathrm{PDR}_{(\mathrm{R1}\rightarrow\mathrm{R2}\,:\,42240\rightarrow42195)} = 99.89\%$; $\mathrm{PDR}_{(\mathrm{R1}\rightarrow\mathrm{R3}\,:\,42240\rightarrow39670)} = 93.92\%$; $\mathrm{PDR}_{(\mathrm{R1}\rightarrow\mathrm{R4}\,:\,42240\rightarrow42227)} = 99.97\%$; $\mathrm{PDR}_{(\mathrm{R1}\rightarrow\mathrm{R5}\,:\,42240\rightarrow41427)} = 98.08\%$; $\mathrm{PDR}_{(\mathrm{R1}\rightarrow\mathrm{R6}\,:\,42240\rightarrow42089)} = 99.64\%$; $\mathrm{PDR}_{(\mathrm{R1}\rightarrow\mathrm{R7}\,:\,42240\rightarrow42129)} = 99.74\%$; $\mathrm{PDR}_{(\mathrm{R1}\rightarrow\mathrm{R8}\,:\,42240\rightarrow42090)} = 99.64\%$; $\mathrm{PDR}_{(\mathrm{R1}\rightarrow\mathrm{R9}\,:\,42240\rightarrow42094)} = 99.65\%$. 92
4.2 Router $\mathrm{R_2} \rightarrow \mathrm{R_d}$ RTT distributions a) within 3σ b) outside 3σ and c) $\mathrm{PDR}_{(\mathrm{R2}\rightarrow\mathrm{R1}\,:\,41951\rightarrow41908)} = 99.90\%$; $\mathrm{PDR}_{(\mathrm{R2}\rightarrow\mathrm{R3}\,:\,41951\rightarrow41919)} = 99.92\%$; $\mathrm{PDR}_{(\mathrm{R2}\rightarrow\mathrm{R4}\,:\,41951\rightarrow41580)} = 99.12\%$; $\mathrm{PDR}_{(\mathrm{R2}\rightarrow\mathrm{R5}\,:\,41951\rightarrow41851)} = 99.76\%$; $\mathrm{PDR}_{(\mathrm{R2}\rightarrow\mathrm{R6}\,:\,41951\rightarrow41758)} = 99.54\%$; $\mathrm{PDR}_{(\mathrm{R2}\rightarrow\mathrm{R7}\,:\,41951\rightarrow41422)} = 98.74\%$; $\mathrm{PDR}_{(\mathrm{R2}\rightarrow\mathrm{R8}\,:\,41951\rightarrow41786)} = 99.61\%$; $\mathrm{PDR}_{(\mathrm{R2}\rightarrow\mathrm{R9}\,:\,41951\rightarrow41696)} = 99.39\%$. 93
4.3 Router $\mathrm{R_3} \rightarrow \mathrm{R_d}$ RTT distributions a) within 3σ b) outside 3σ and c) $\mathrm{PDR}_{(\mathrm{R3}\rightarrow\mathrm{R1}\,:\,42228\rightarrow39993)} = 94.71\%$; $\mathrm{PDR}_{(\mathrm{R3}\rightarrow\mathrm{R2}\,:\,42228\rightarrow42200)} = 99.93\%$; $\mathrm{PDR}_{(\mathrm{R3}\rightarrow\mathrm{R4}\,:\,42228\rightarrow41282)} = 97.76\%$; $\mathrm{PDR}_{(\mathrm{R3}\rightarrow\mathrm{R5}\,:\,42228\rightarrow42186)} = 99.90\%$; $\mathrm{PDR}_{(\mathrm{R3}\rightarrow\mathrm{R6}\,:\,42228\rightarrow42166)} = 99.85\%$; $\mathrm{PDR}_{(\mathrm{R3}\rightarrow\mathrm{R7}\,:\,42228\rightarrow42053)} = 99.59\%$; $\mathrm{PDR}_{(\mathrm{R3}\rightarrow\mathrm{R8}\,:\,42228\rightarrow42053)} = 99.59\%$; $\mathrm{PDR}_{(\mathrm{R3}\rightarrow\mathrm{R9}\,:\,42228\rightarrow41986)} = 99.43\%$. 94

4.4 Router $R_4 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c) $PDR_{(R4\rightarrow R1:42670\rightarrow42641)} = 99.93\%$; $PDR_{(R4\rightarrow R2:42670\rightarrow42284)} = 99.10\%$; $PDR_{(R4\rightarrow R3:42670\rightarrow41578)} = 97.44\%$; $PDR_{(R4\rightarrow R5:42670\rightarrow42650)} = 99.95\%$; $PDR_{(R4\rightarrow R6:42670\rightarrow37104)} = 86.96\%$; $PDR_{(R4\rightarrow R7:42670\rightarrow42649)} = 99.95\%$; $PDR_{(R4\rightarrow R8:42670\rightarrow42489)} = 99.58\%$; $PDR_{(R4\rightarrow R9:42670\rightarrow42513)} = 99.63\%$. 95

4.5 Router $R_5 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c) $PDR_{(R5\rightarrow R1:42395\rightarrow41646)} = 98.23\%$; $PDR_{(R5\rightarrow R2:42395\rightarrow42293)} = 99.76\%$; $PDR_{(R5\rightarrow R3:42395\rightarrow42334)} = 99.86\%$; $PDR_{(R5\rightarrow R4:42395\rightarrow42372)} = 99.95\%$; $PDR_{(R5\rightarrow R6:42395\rightarrow42340)} = 99.87\%$; $PDR_{(R5\rightarrow R7:42395\rightarrow38878)} = 91.70\%$; $PDR_{(R5\rightarrow R8:42395\rightarrow41875)} = 98.77\%$; $PDR_{(R5\rightarrow R9:42395\rightarrow42179)} = 99.49\%$. 96

4.6 Router $R_6 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c) $PDR_{(R6\rightarrow R1:42756\rightarrow42636)} = 99.72\%$; $PDR_{(R6\rightarrow R2:42756\rightarrow42602)} = 99.64\%$; $PDR_{(R6\rightarrow R3:42756\rightarrow42677)} = 99.82\%$; $PDR_{(R6\rightarrow R4:42756\rightarrow37269)} = 87.17\%$; $PDR_{(R6\rightarrow R5:42756\rightarrow42708)} = 99.89\%$; $PDR_{(R6\rightarrow R7:42756\rightarrow40755)} = 95.32\%$; $PDR_{(R6\rightarrow R8:42756\rightarrow42671)} = 99.80\%$; $PDR_{(R6\rightarrow R9:42756\rightarrow42602)} = 99.64\%$. 97

4.7 Router $R_7 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c) $PDR_{(R7\rightarrow R1:42608\rightarrow42491)} = 99.73\%$; $PDR_{(R7\rightarrow R2:42608\rightarrow42039)} = 98.66\%$; $PDR_{(R7\rightarrow R3:42608\rightarrow42438)} = 99.60\%$; $PDR_{(R7\rightarrow R4:42608\rightarrow42579)} = 99.93\%$; $PDR_{(R7\rightarrow R5:42608\rightarrow38682)} = 90.79\%$; $PDR_{(R7\rightarrow R6:42608\rightarrow40591)} = 95.27\%$; $PDR_{(R7\rightarrow R8:42608\rightarrow42551)} = 99.87\%$; $PDR_{(R7\rightarrow R9:42608\rightarrow42531)} = 99.82\%$. 98

4.8 Router $R_8 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c) $PDR_{(R8\rightarrow R1:41901\rightarrow41762)} = 99.67\%$; $PDR_{(R8\rightarrow R2:41901\rightarrow41731)} = 99.59\%$; $PDR_{(R8\rightarrow R3:41901\rightarrow41736)} = 99.61\%$; $PDR_{(R8\rightarrow R4:41901\rightarrow41742)} = 99.62\%$; $PDR_{(R8\rightarrow R5:41901\rightarrow41413)} = 98.84\%$; $PDR_{(R8\rightarrow R6:41901\rightarrow41823)} = 98.81\%$; $PDR_{(R8\rightarrow R7:41901\rightarrow41866)} = 99.92\%$; $PDR_{(R8\rightarrow R9:41901\rightarrow41825)} = 99.82\%$. 99

4.9 Router $R_9 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c) $PDR_{(R9 \rightarrow R1\,:\,41881 \rightarrow 41748)} = 99.68\%$; $PDR_{(R9 \rightarrow R2\,:\,41881 \rightarrow 41660)} = 99.47\%$; $PDR_{(R9 \rightarrow R3\,:\,41881 \rightarrow 41671)} = 99.50\%$; $PDR_{(R9 \rightarrow R4\,:\,41881 \rightarrow 41754)} = 99.70\%$; $PDR_{(R9 \rightarrow R5\,:\,41881 \rightarrow 41708)} = 99.59\%$; $PDR_{(R9 \rightarrow R6\,:\,41881 \rightarrow 41733)} = 98.65\%$; $PDR_{(R9 \rightarrow R7\,:\,41881 \rightarrow 41828)} = 99.87\%$; $PDR_{(R9 \rightarrow R8\,:\,41881 \rightarrow 41794)} = 99.79\%$. 100
4.10 Mesh network route performance by *k-means* clustering analysis. 101
4.11 Mesh network route reliability by regression analysis. 102
4.12 Router RTTs at various throughputs. 106

List of Tables

2.1 Parameters and settings obtained from a) Online Power Profiler [24] estimating the lowest b) average power consumption of a slave's advertisement over a 10245 ms interval with a BLE event of 4.36 ms. 27

3.1 Links displaying the average signal strength with link cost between interconnected routers forming the mesh network. 83

4.1 Router $R_1 \rightarrow R_d$ routing table. 109

4.2 Router $R_2 \rightarrow R_d$ routing table. 109

4.3 Router $R_3 \rightarrow R_d$ routing table. 109

4.4 Router $R_4 \rightarrow R_d$ routing table. 110

4.5 Router $R_5 \rightarrow R_d$ routing table. 110

4.6 Router $R_6 \rightarrow R_d$ routing table. 110

4.7 Router $R_7 \rightarrow R_d$ routing table. 111

4.8 Router $R_8 \rightarrow R_d$ routing table. 111

4.9 Router $R_9 \rightarrow R_d$ routing table. 111

List of Equations

2.1 Gaussian pulse shape filter. 47
2.2 Received signal strength versus distance estimation. 48
2.3 CPFSK modulation of baseband signal over carrier frequency. 48
2.4 Frequency shift keying modulation index. 48
2.5 Modulated frequency deviation. 49
2.6 Carrier signal spread over channels. 49
2.7 Q-OPSK modulator phases representing binary data sequences. 61
2.8 Half-sine pulse shaping. 61
2.9 O-QPSK modulation output signal. 64
3.1 Log-distance model. 79
4.1 Number of standard deviations from mean. 88
4.2 Mean of RTT vector. 90
4.3 Standard deviation of RTT vector. 90
4.4 Gaussian kernel function. 90
4.5 Packet delivery ratio. 91
4.6 Bellman-Ford in DVR. 108

Acronyms and Abbreviations

ABS	Acrylonitrile Butadiene Styrene
APP	Application
ASIC	Application Specific Integrated Circuit
API	Application Programming Interface
ATLAS	A Toroidal LHC ApparatuS
aQFN	advanced Quad Flat No-lead
BFSK	Binary Frequency Shift Keying
BPSK	Binary Phase Shift Keying
BLE	Bluetooth Low-Energy
CERN	European Organisation of Nuclear Research
CMS_1	Compact Muon Solenoid
CMS_2	Central Management System
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
DC_1	Direct Current
DC_2	Duty Cycle
DVR	Distance Vector Routing
DK	Development Kit
DLL	Data Link Layer
EEMBC	EDN Embedded Microprocessor Benchmark Consortium
EDR	Enhanced Data Rate

FDMA	Frequency Division Multiple Access
FHSS	Frequency Hop Spread Spectrum
FS	Frequency Synthesiser
GCC	GNU Compiler Collection
GFSK	Gaussian Frequency Shift Keying
GPF	Gaussian Pulse Filter
GPS	Global Positioning Satellite
HEPP	High Energy Particle Physics
HTEL	High-Throughput Electronics Laboratory
HCI	Host Controller Interface
IC	Integrated Circuit
IED	Intelligent Embedded Device
ICMP	Internet Control Message Protocol
IDE	Integrated Development Environment
ICPP	Institute for Collider Particle Physics
IEEE	Institute of Electrical and Electronic Engineers
IoT	Internet-of-Things
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
ISA	Industrial, Scientific and Medical
ISI	Inter Symbol Interference
LHC	Large Hadron Collider
LLC	Logical Link Control
LoRaWAN	Long Range Wide Area Network
NA	Not Available
MAC	Medium Access Control
NRZ	Non-Return Zero
MCU	Micro-Controller Unit
MLE	Mesh Link Establishment
OEM	Original Equipment Manufacturer

OSI	Open System Interconnectivity
O-QPSK	Offset-Quadrature Phase Shift Keying
PHY	Physical Layer
PLL	Phase-Locked Loop
PCB	Printed Circuit Board
PCD	Portable Computer Device
PDF	Probability Density Function
QoS	Quality-of-Service
QPS	Quench Protection System
RF	Radio Frequency
RAM	Random-Access Memory
ROM	Read-Only Memory
RSIC	Radio-Slave Interface Circuit
RSSI	Received Signal Strength Indicator
RTT	Round-Trip Time
SDK	Software Development Kit
SIG	Special Interest Group
SoC	System-on-Chip
SPI	Serial Peripheral Interface
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
VCO	Voltage-Controlled Oscillator
WLCSP	Water-Level Chip-Scale Packages
WorldFIP	World Factory Instrumentation Protocol
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Chapter 1

Introduction

The High-Throughput Electronics Laboratory (HTEL) within the Institute for Collider Particle Physics (ICPP) at the University of the Witwatersrand is working in conjunction with the European Organisation of Nuclear Research (CERN) in Internet-of-Things (IoT) solutions using state-of-the-art low-power consumption wireless technologies. In this research, low-power consumption is defined as small wireless devices that are powered from pocket-sized batteries while throughput means the rate at which data can be reliably sent and received between devices for communications in a network. These small wireless devices can be connected to the internet for various applications in many industries and remotely monitored and controlled from anywhere around the world. The upcoming sections discuss two instrumentation applications and outline the importance of IoT in industry. The deployment of IoT solutions at the Large Hadron Collider (LHC)[1] is, however, in the infancy phase and currently being researched [1]. For this reason, an application is presented that focus on a protection system currently deployed at the LHC and proposes

[1] https://home.cern/science/accelerators/large-hadron-collider.

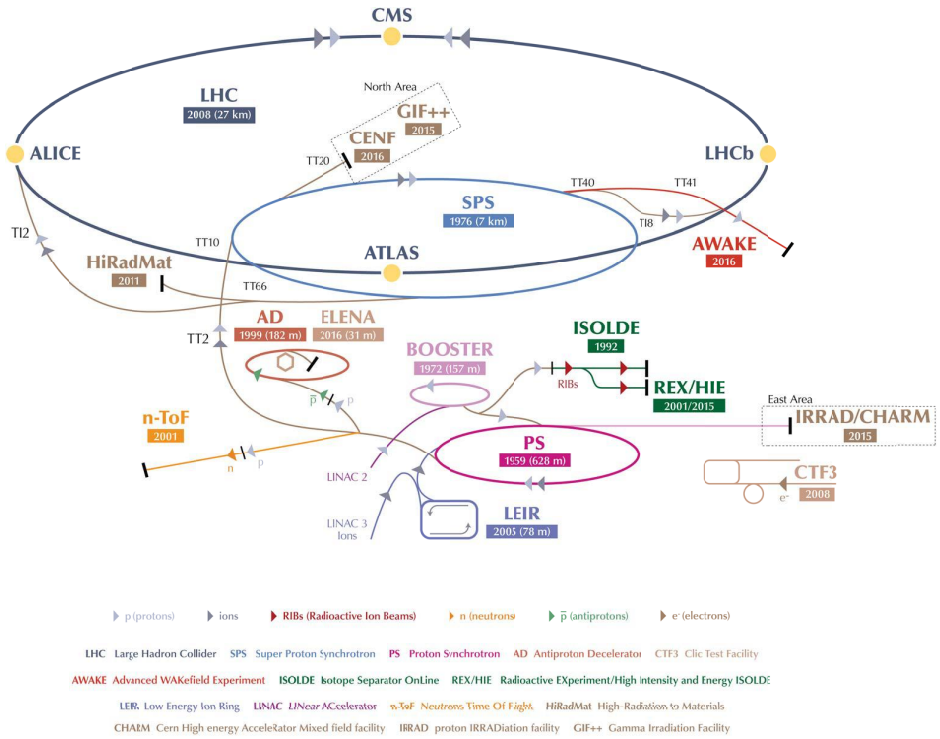


Figure 1.1: CERN's accelerator complex (Image by CERN).

an IoT solution for low-throughput wireless data communications. Finally, a second application is presented of research currently being conducted at the HTEL that focusses on potential IoT anti-theft solutions in South Africa.

1.1 LHC at CERN

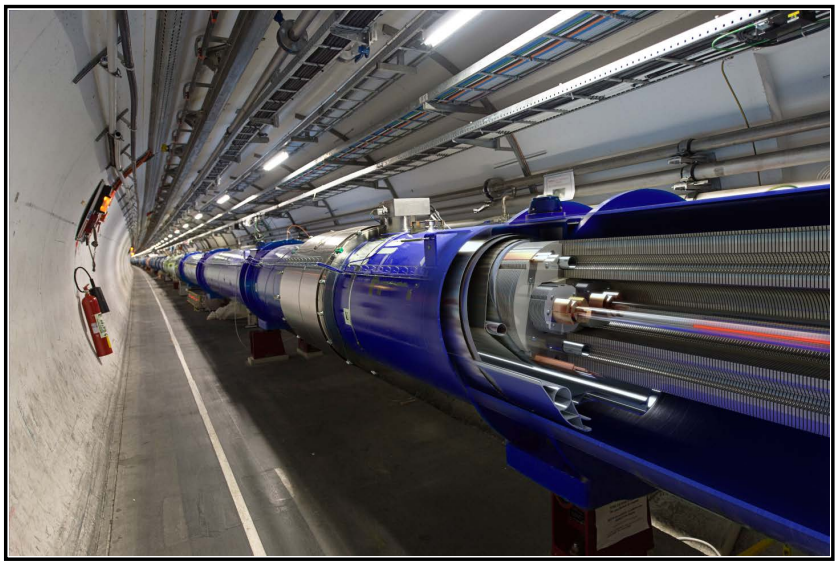
The LHC [2], located on the Franco-Swiss border near Geneva, Switzerland, is the world's most powerful particle accelerator and the most significant part of the CERN accelerator complex. The LHC is located approximately 100-meters underground and consists of a 27-kilometre ring of 1232 main dipole and 392 main quadrupole superconducting elec-

tromagnets interconnected to make the largest machine ever built. The superconducting magnets form an intricate part of the LHC and are responsible for the stable trajectory and alignment of the beams. The LHC has 16 Radio Frequency (RF) superconductive cavities driven each by a high-power klystron. The klystron is intensity-modulated at a frequency of 400 MHz (resonating inside the cavity) to accelerate the particles and ensure particles travel in bunches at the desired energy.

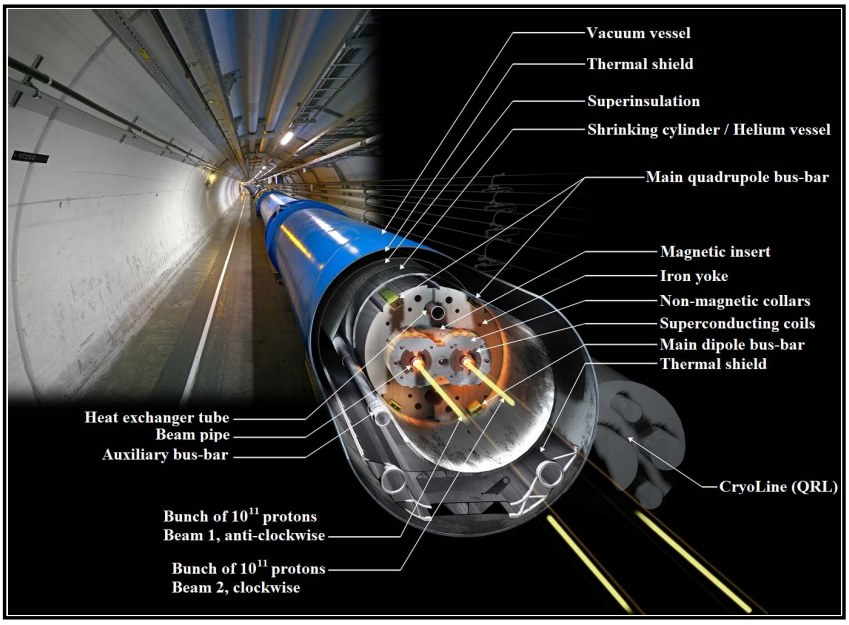
The accelerator complex illustrated in **Figure 1.1**, requires that proton particles be removed from hydrogen atoms by a duoplasmatron. The protons are then accelerated by the LINear ACcelerator (LINAC 2) for injection into the Proton Synchrotron Booster (PSB), Proton Synchrotron (PS), Super Proton Synchrotron (SPS) and finally the LHC where the energy of the protons reaches more than 99.99% the speed of light. The two beams made out of protons travel each through an ultra-high vacuum (1.013×10^{-10} mbar) pipe guided around the ring by the superconducting magnets. The ultra-high vacuum pipes prevent collisions with gas molecules in order to accelerate the protons. The beams are then made to collide at 6.5 TeV each to create sub-atomic particles at four locations where the A Toroidal LHC ApparatuS (ATLAS), Compact Muon Solenoid (CMS), A Large Ion Collider Experiment (ALICE) and Large Hadron Collider beauty (LHCb) particle detectors are located. Events of interest out of one billion collisions per second are captured by sophisticated data acquisition electronics and logged for analysis.

1.1.1 LHC superconductor magnets

A single superconducting magnet (hereafter referred to as the main dipole) shown in **Figure 1.2a** is 15 meters long and weighs 35-tonnes. The main dipole consists of a pair



(a) Superconductor magnet located in the LHC tunnel.



(b) Superconductor magnet electro-mechanical view.

Figure 1.2: LHC dipole depict in a) a three-dimensional view of the superconductive cable surrounding two proton beam pipes travelling in opposite directions and b) the cross-section of the 15-meter cryodipole with component descriptions (Images by CERN).

of superconducting coils depicted in **Figure 1.2b** that are in essence oppositely charged magnetised poles surrounding the beam pipes. A sophisticated Niobium Titanium (NbTi) Rutherford wiring scheme is used for the main dipole (for bending the beam), the dipole-quadrupole (for focussing of beam) and sextupoles (for error correction of the beam) magnets and cooled down to 1.9 K (-271.3 °C) in order to generate the high magnetic field [3]. The NbTi wires become super-conductive at 1.9 K and virtually offer zero resistance to the flow of electrical current in the magnets. The main dipole, dipole and quadrupole magnets are connected in series with specialised superconducting bus-bars that uses a dedicated splicing technique to interconnect the magnets. The current carrying capacity of the main dipole is 13000 A DC_1 that flows through the interconnected dipoles to create a magnetic field of 8.3 T[2] to bend and guide the 6.5 TeV proton beams around the ring. The accelerator is equipped with the largest cryogenic helium gas plant in the world to provide a refrigeration loop to the magnets around the LHC ring. An accelerator layout of a single LHC cell with dipoles, quadrupoles and the corrector magnets is illustrated in Sonnemann [4, p.9].

1.1.2 LHC protection system

A Quench Protection System (QPS) is installed underneath the main dipoles in the LHC tunnel. The QPS electronics protects the LHC superconducting elements entering a resistive state when powered-on and consequently leading to a thermal runaway [5, p.1725]. The QPS monitors signals received from the superconducting elements where 4000 internal

[2]The LHC's magnetic field is thousands of times more powerful than the earth's magnetic field.

trigger channels require a healthy state to power the dipoles in the LHC [6]. A non-healthy state means that a potential quench may exist. Once a quench is detected, power converters are shut down, quench heaters are enabled that bypasses the current flowing through the magnet via a parallel diode where the energy is discharged through dump resistors. An instrumentation feed-through located on the interconnection part of the main dipole shown in **Figure 1.2b** is provided utilising bridge detectors. Wiring employing voltage taps from the bridge detectors installed on the magnets are used to measure current and voltages [4, p.166-167]. These analog input signals are used as inputs to the QPS from the superconductor magnet to trigger an interlock system to dump the beams safely and stop the accelerator. A schematic view of a dipole circuit implementing quench detection and protection for 154 main dipoles is illustrated in Rossi [3]. Each QPS unit connects the instrumentation of four main dipoles. The QPS units communicate to a supervisory workstation using a fieldbus network known as WorldFIP where data is stored temporarily in a local storage system [5, p.1727]. WorldFIP is an open Access European standard (EN50170) and international fieldbus protocol [7]. The continuous-time data gathered from the QPS instrumentation include the following inputs:

1.) $U_{(DIFF)}$: Voltage measured between terminals of the main dipole.

2.) $I_{(DCCT)}$: Direct Current-Current Transformer (DCCT) that measures the current flowing through the main dipole.

3.) $U_{(RES)}$: Resistance calculation from voltage $U_{(DIFF)}$ and current $I_{(DCCT)}$.

4.) $I_{(DIDT)}$: Time-derivative calculation of current $I_{(DCCT)}$.

The research by Wielgosz et al. [8, p.9-11] fed the inputs to an anomaly detection algorithm (deep Learning RNN models with dedicated data compression scheme) to predict

a quenched state that can cause severe damage in the main dipole. The algorithm was tested off-line by means of an Intelligent Embedded Device (IED) using the data stored on the CERN Accelerator Logging Service (CALS). An IED is a Printed Circuit Board (PCB) that embeds an advanced low-power consumption integrated Micro-Controller Unit (MCU) with a radio transceiver on a single System-on-Chip (SoC) capable of running complex algorithms. Instead of using a fully analog electronics hardware approach, computationally intensive algorithms can be implemented online at a much faster rate on the CALS (with some latency delay in telecommunications) and/or an IED to trigger the power converter shutdown and activate the particular QPS.

1.1.3 Concept of IoT at the LHC

Low-power and cost-effective wireless mesh networks in the LHC tunnel would provide multiple paths for telecommunications between potentially hundreds of interconnected nodes. Nodes may connect to almost any kind of instrumentation or embedded system containing various types of sensors and/or actuators. A gateway allows control and monitoring of the interconnected nodes in the network for IoT connectivity and cloud computing. Cloud computing reduces costs in storing large data sets and provides analytical tools to analyse data. For example, events are triggered from the inputs outputed from the QPS sensors presented in **section 1.1.2** managed from the IED and transmitted through the mesh and stored on the cloud, followed by an analysis by the data scientist using analytical tools that provide feedback to the intelligent device. The collection of vast amounts of data obtained from sensors on critical machinery at CERN's LHC radiation controlled areas, enables

physicists and engineers to accurately predict without the need for physical inspection, the operational lifespan of the machine and ultimately predict unplanned failures. Special care has to be taken regarding node locations and antenna orientations in the LHC tunnel. The design specifications of the IED must be able to operate near the main dipole where radio wave propagation is possible in all directions. Main dipoles will emit radiation and strong magnetic fields and further research is needed regarding ISM RF propagation, latency delays between hops and a radiation tolerant electronics design for deployment in the LHC tunnel.

Wireless mesh networks and IoT connectivity are not restricted to the instrumentation in main dipoles but can be used for almost any low-throughput (in terms of wireless data communications) monitoring and control applications in the LHC tunnel. Typical applications include radiation monitoring, gas detection monitoring, humidity detection systems, location tracking of CERN personnel and safety protection systems. Long Range Wide Area Network (LoRaWAN)[3] is a commercially available sub-GHz long-range low-power wireless technology using a star-of-stars network topology that is currently the selected IoT wireless technology used at CERN [9, 10].

1.1.4 Conclusion

The industrial IoT can provide large organisations such as CERN with monitoring and control capabilities of machinery by means of instrumentation that is connected through the internet to a cloud storage system. Big data tools developed at CERN can analyse

[3] https://lora-alliance.org/about-lorawan.

large data sets gathered from the instrumentation to conduct statistical analytics or machine learning to make informed decisions to optimise operational processes and/or protect critical equipment. Instrumentation in the field may be connected to an IED and have computationally intensive algorithms implemented that trigger IoT events of interest sent to the cloud, evaluated and the analysed data fed back to the IED or control centre. Consequently, big data can be stored on the cloud and analysed either online or offline by the data scientist to predict future outcomes in operational processes. The potential applications by utilising a low-power Wireless Sensor Network (WSN) at CERN is extremely diverse and may be particular useful in control systems and condition monitoring of particle physics instrumentation systems.

1.2 The HTEL anti-theft initiative

In 2015, the South African government launched a multi-billion ZAR project to modernise educational institutions by the year 2020. Reports from The Citizen [11] and Business Tech [12] indicated that with the initial roll-out, a substantial amount of tablets[4] were lost due to theft that consequently led to the withdrawal of 88000 tablets by the Gauteng Department of Education (GDE). As a result, the tablets were fitted with tracking devices that use Global Positioning Satellites (GPS). One of the significant shortfalls of the anti-theft system upgrade was that an external tracking device fitted to the tablet could only be tracked when powered by the tablet but not when the tablet is switched off.

[4]Tablets are referred to computers used for educational purposes running either Android, iOS or Windows operating system where iOS and Windows are manufactured-build proprietary operating systems.

Furthermore, stolen tablets have the risk of the firmware being modified which may render the tablet's internal location services useless. Consequently, a new operating system may be installed and stolen tablets can be sold on the black market. Adversely, the GDE has proven to have limited resources in terms of investigations teams in retrieving stolen tablets. The independent investigation in [13] lead by the HTEL, discovered that schools in rural areas around the country to be predominantly targeted for the theft of tablets. With these ongoing challenges to protect educational devices, a strong need followed to develop security measures locally to safeguard tablets against theft in learning institutions.

1.2.1 PCD anti-theft solutions

A Portable Computer Device (PCD) is referred to as a smart-device such as a tablet that can communicate over the internet. Apple is an international technology company that design and develops smart-devices running its proprietary iOS operating system. The company provides an iCloud service where users can remotely update device firmware and enable or disable security features for their devices. Their proprietary software application, 'Find my iPad', interface with GPS that allows device tracking. To enable tracking and locate a device, an Apple user is required to sign into the iCloud service on a personal computer. The device being tracked has to be powered on and connected to the internet for the tracking system to work. Devices are protected from unauthorised access by 'Passcode' verification which is a security feature used to protect a user's data. In later iOS 7 versions and upwards, the 'Activation Lock' feature was introduced. This security feature which is automatically enabled when 'Find my iPad' is activated, provides a powerful deterrent

in the stealing of devices. When activated prior to when a device is lost or stolen, the feature requires an Apple ID and password to access the device, even after a factory reset is performed. The security feature according to Activation Lock [14], makes any iOS device almost impossible to access without the correct Apple ID and password. Devices also allow a 'Recovery Mode' that may be entered into manually by a user connected to iTunes [15]. This allows the user to restore the device to the factory default setting. Later iOS 11.4.1 devices require a 'Passcode' entry to access the device through a USB cable connected to a PC running the iTunes software. On older Apple A10, A11, S3 SoCs, devices can bypass the 'Passcode' protection function and be booted from a 'Recovery Mode' to restore the firmware to factory settings. On the older iOS versions, too many incorrect 'Passcode' entries lead to deletion of the device data [16]. This means without 'Activation Lock' enabled, a device can be recovered to remove the 'Passcode' by performing a system restore using iTunes.

Android products provides open access to their operating systems which means that custom code may be written on the ROM of the devices by companies or developers. Read-Only Memory (ROM) in smart-devices is generally referred to as flashable non-volatile memory that contains machine code from an image file of the Android APPs, operating system and firmware. Third-party APPs are available for download on the Google Play Store[5] service that requires a GPS and an internet connection to track a device. Similarly to iOS, the device being tracked has to be powered on and connected to the internet for the tracking system to work. The device can be flashed with custom code on the ROM

[5]The Google Play Store serves as an official APP store for users of the Android operating system. Developers develop APPs using the Android Studio IDE coded in Java and XML.

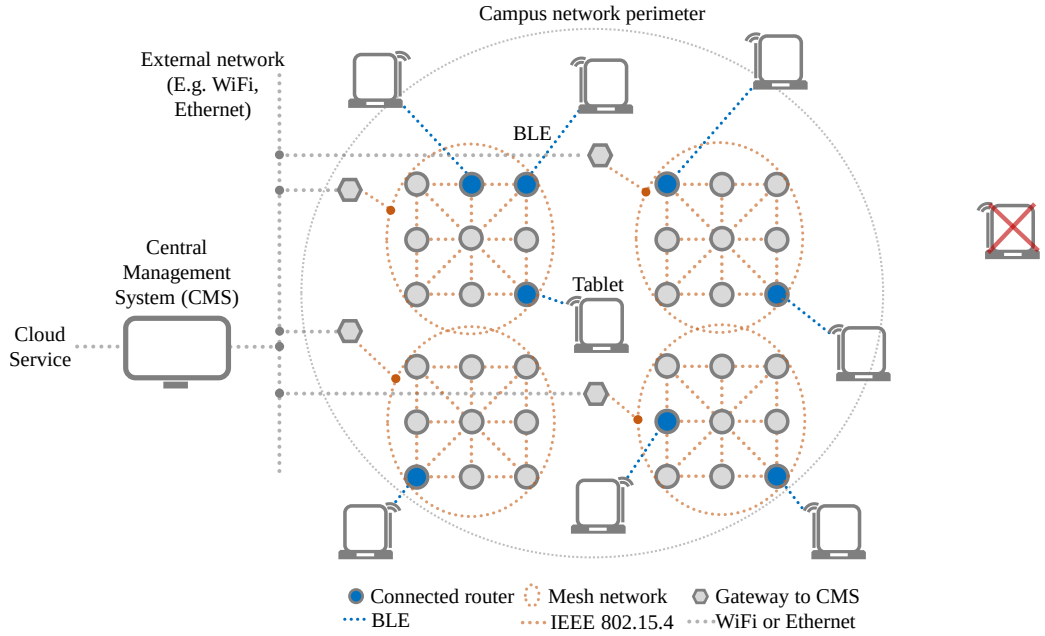


Figure 1.3: Proposed HTEL anti-theft solution where tablets connect to the nearest node based on their location in multiple mesh networks.

when stolen rendering any anti-theft APPs useless and the device being used again.

Third-party anti-theft solutions are currently available to protect devices against theft. One of the predominantly used security solutions available on the market and developed by a Canadian company, called Absolute, is the patented 'Persistence' technology [17]. It is marketed by NetTrace in South Africa that provides advanced security features to computer devices for theft recovery. The 'Persistence' firmware enables tracking software and is embedded in most major Original Equipment Manufacturers (OEM) during the manufacturing process. The activation of the firmware on the device requires a licence and is annually renewed. The firmware prevents the installation of custom operating systems on the ROM of the device and prevents the deletion of APPs by using the self-healing 'Persistence' technology. Once a device is registered with Absolute and activated,

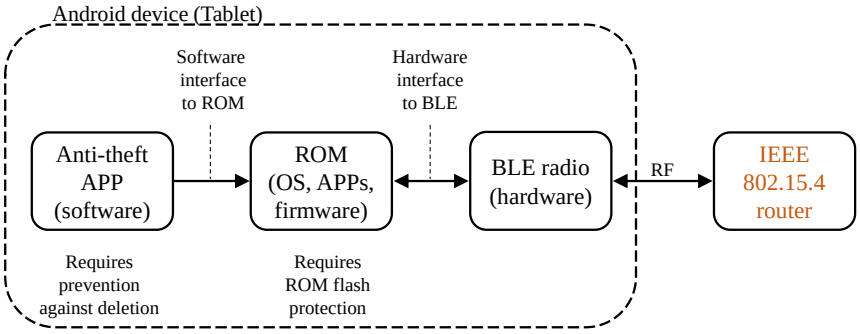


Figure 1.4: Block diagram illustrating hardware and software systems for data communications from an Android device to the nearest node in an IEEE 802.15.4 based mesh network.

it regularly communicates with the Absolute monitoring services. This is followed by a custom agent script that is retrieved from the Absolute servers and executed on the device. Critical APPs are protected by the script that regularly checks their presence on the device and if undetected, APP components are downloaded and installed again. For example, a Samsung device running the Android operating system may have a custom developed APP protected by Absolute's professional's services that can be configured to use the self-healing technology [18]. If the ROM is flashed or the APP is deleted, it is automatically reinstalled by the 'Persistence' firmware as soon as there is an internet connection. The anti-theft solutions may, however, be very expensive for incorporating a large number of devices throughout educational institutions. The 'Persistence' technology requires the internet for the 'Persistence' firmware to work and a GPS to detect and aid in the recovering of stolen devices.

1.2.2 Proposed IoT at the HTEL

The HTEL[6] is working towards developing a solution to the problem of tablet theft by effectively deterring thieves and organised crime from stealing educational devices, and electronics in general. An anti-theft system is proposed that use low-power mesh networks that may operate independently of other networks for data communications to tablets. Devices that operate outside the working perimeter of these independent ad-hoc networks will initiate a shutdown of the tablet's operating system. The proposed anti-theft system is depicted in **Figure 1.3** that consists of multiple mesh networks that cover large areas in an educational institution. When a tablet is powered on, an Android APP running on the tablet will require the tablet to connect to the nearest node and send information to a Central Management System ($CMS_{\mathbf{2}}$) through the mesh network. Failure of the APP to connect to any of the nodes in the mesh network will initiate a shutdown of the device's operating system. A comprehensive technical overview of the wireless technologies employed in the proposed HTEL anti-theft solution is presented in **Chapter 2**.

The block diagram depicted in **Figure 1.4**, illustrates the ROM (where the firmware is located) interface to communicate to the Android APP and Bluetooth Low Energy (BLE) hardware. In order to shut down a tablet, the firmware communicates to the BLE hardware which receives the RF signals from the connected node. The Android application may use a polling algorithm to check the data received from the node, which will be ultimately responsible in deciding whether the tablet should be shut down or be rendered inoperable. For this reason, the PCD will require upgraded firmware or the activation of third-party firmware already embedded on the PCD firmware to prevent ROM flashing and protect

[6] http://hep.wits.ac.za/research/electronics.php.

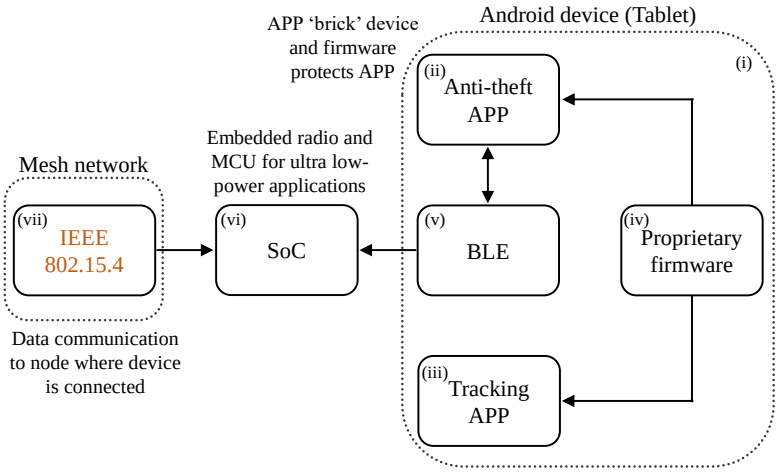


Figure 1.5: Block diagram of proposed HTEL anti-theft solution illustrating main technology systems.

the Android APP against deletion. In embedded systems, the firmware is software that is permanently programmed into flashable ROM by the developer that communicates to the hardware of the device. When referring to smart-devices, such as tablets, Android firmware on a tablet refers to the firmware, operating system and APPs. IoT connectivity may be added providing important tablet information for monitoring and control to a CMS_2, by connecting the mesh networks through a redundant gateway scheme. The HTEL is currently in engagements with a reputable tablet manufacturer regarding the firmware developments on the device. Alternatively, similar third-party Android firmware such as found by Absolute [17] may be developed or activated to prevent the flashing of the device operating system and protect APPs against deletion.

The main technology components displaying the state-of-the-art in **Figure 1.5** of the HTEL proposed anti-theft system consist of the following:

i.) The tablet running an Android operating system.

ii.) An HTEL developed APP installed on the device used primarily to communicate to the mesh network via BLE and initiate a shutdown of the device. The anti-theft APP, similar to the 'Activation Lock', is responsible for rendering the device inoperable by denying the user access to the device when used outside the network coverage of the mesh network.

iii.) A tracking APP installed on the device can be used to track the device if the device is powered on and connected to the internet. The APP may be developed by the HTEL or acquired using a third-party solution. The tracking APP runs independently of the mesh network and $\mathbf{CMS_2}$.

iv.) The proprietary firmware similar to 'Persistence' that protects the operating system and APPs.

v.) The wireless technology used for point-to-point data communications from the device to the nearest node in the mesh network. Smart-devices such as tablets incorporate a computer processor and radio that natively support BLE formerly known as Bluetooth Smart.

vi.) An SoC, embedding an MCU and RF transceiver capable of running multiple low-power network protocols concurrently (e.g. IEEE 802.15.4 and BLE).

vii.) Finally, the proposed low-power IEEE 802.15.4 wireless mesh network.

1.2.3 Conclusion

Devices running the iOS operating system have shown superior security features compared to Android. However, iOS is a proprietary operating system and smart-devices running

these operating systems are considerably more expensive. Android provides cost-effective smart-devices and users have the freedom to customise or develop their own operating systems making it the most used operating system in the world. Third-party firmware such as 'Persistence' from Absolute [17] may be activated on smart-devices that considerably enhance the security of the Android device. The advantages and disadvantages of proprietary and non-proprietary anti-theft solutions (iOS, Android and Absolute) are presented in **Appendix I**.

The HTEL proposed anti-theft system was discussed and the main technology components were introduced. Unique to other security solutions, the HTEL anti-theft system uses low-power and cost-effective ad-hoc wireless mesh network to monitor and control tablets. The HTEL proposed anti-theft solution may act as a strong deterrent of tablet theft and may be potentially found feasible for implementation in educational institutions and other industries of South Africa. IoT connectivity is possible allowing tablets to remotely connect to the anti-theft APP where for example the tracking data of connected devices may be stored on the cloud for further processing. The HTEL anti-theft system may make use of online data analysis to conduct anomaly detection and tracking of devices in the network. The proposed anti-theft solution is however not restricted to the monitoring of tablets but to any low-power instrumentation control and monitoring application that requires wireless mesh communications to a $\mathbf{CMS_2}$.

1.3 Problem Statement

This research focuses on a mesh network design and development using a particular low-power wireless technology consisting of embedded hardware and software developments. The following sub-problems outline the problem statement.

1.3.1 Low-power wireless technologies

A network needs to be developed that allows data communication between small and low-power consumption devices in a mesh topology with IoT capabilities. A cost-effective SoC that houses an embedded MCU and an RF transceiver from a reputable silicon chip manufacturer is required to execute the software stack of the wireless mesh technology and allow communication with a smart-device. The researcher has to determine the Quality of Service (QoS) or, more specifically, analyse the performance and reliability metrics between the interconnected nodes that form the mesh network. In this study, network performance deals with the latency and the hop distances that a packet travels from source to a destination node, while network reliability determines whether or not a packet is successfully delivered. It is, therefore, imperative to understand these metrics of the mesh network before deployment in monitoring and control applications.

1.3.2 Mesh network development

The location and distance between the interconnected nodes that form a mesh in a heavily obstructed indoor environment will form the network test-bench. Hardware and software

development are required to set up the test-bench and emulate a busy network in which each node in the mesh continually transmits packets to other nodes in sequence. The data analysis process requires logging capabilities without user intervention on each node in the network, which will run for multiple days and log performance and reliability metrics.

1.3.3 Mesh network characterisation

The packet transmissions between interconnected nodes that form the mesh can exceed millions of packets. The independent ad-hoc network is characterised to understand any related performance and reliability issues.

1.4 Hypothesis

Low-power RF devices consisting of a single-link for point-to-point communications are limited in terms of transmission distances. A single-link is referred to as a data communication route that consists of a single hop between only two nodes, while multiple links refer to the use of more than one interconnected node to relay a packet from a source to a destination node. Packets may be routed through multiple links known as the mesh through multi-hopping and increase communication distances.

In a network consisting of multiple nodes, packets may be transmitted from the source and destination node where latency and packet loss are logged to determine the performance and reliability of the network. Data are analysed by studying the round-trip time distributions using statistical and/or machine learning techniques, which may provide in-

formation about multi-hopping distances without the need to simulate the routing algorithm of the particular wireless mesh technology. High-throughput data communications and greater multi-hopping distances will cause higher latency delays. Harsh building environments may cause fluctuating signal strengths due to reflections and scattering due to obstacles between interconnected nodes. This may cause dynamic routing by the routing protocol, consequently leading to higher latency delays and packet loss. Similarly, other wireless devices operating on similar RF bands may add additional network traffic in parts of the building, which may potentially lead to packet collisions and adversely affect the performance and reliability of the network.

1.5 Methodology

An SoC with a reputable silicon chip manufacturer capable of embedding an IoT wireless technology will be fully utilised during the inception phase of network development. The network will consist of nine nodes, without a $\mathbf{CMS_2}$ and distributed in a harsh indoor building environment. The research will consist of three experimental parts:

1.) The first part of the experimental study will focus on modelling signal strength with respect to various distances between two nodes in the Physics building ground floor to determine RF coverage. Results from the experiment will aid in the distribution of the nodes in the building.

2.) In the second part of the experimental study, a network test-bench will be developed, as illustrated by **Figure 1.3**, where unicast Internet Control Message Protocol (ICMP) requests from a single source R_s to multiple destination nodes R_d are con-

ducted sequentially. Unicast messaging is data communications between a single source and destination node that may contain multiple links. Only a single source node will be transmitting packets.

3.) Similarly, in the third part of the experimental study, unicast ICMP requests from a single source R_s to multiple destination nodes R_d are conducted sequentially over an extended period of time. The difference being that source node R_s is randomly activated remotely to start running an ICMP algorithm and emulate a busy mesh network. Instead of a single source node transmitting, all the source nodes will be transmitting packets in the network. Similar methodologies in big data analysis as found in particle physics will be adopted to analyse and characterise the network performance and reliability. The results obtained from the statistical analysis are compared with a simulation of the wireless technology routing protocol to confirm hop distances packets travelled.

1.6 Importance of Research

This research aims to develop a low-power wireless mesh network for data communications between interconnected nodes. The HTEL proposes a locally produced solution to tablet theft and avoids incorporating commercial, off-the-shelf, expensive anti-theft solutions. The latest IEEE 802.15.4 low-power wireless technologies in IoT connectivity will be explored and analysed. This can lead to the future development of innovative applications that require low-throughput data communications through mesh networks to cloud storage and/or computing services.

1.7 Research Delineation of the ICPP

This research forms part of a proposal submitted to the Department of Science and Technology and initiated by the HTEL at the University of the Witwatersrand. The work conducted in this study incorporates the IEEE 802.15.4 low-power communications standard, leading to the development of a wireless mesh technology with IoT capabilities. The limitations of this research are outlined and will be dealt with in future studies:

i.) The study of RF propagation and attenuation in an indoor environment, meant to determine the causes of network degradation, such as the effects that obstacles and RF sources have on wireless communications.

ii.) Data communications of devices through a gateway for monitoring and control from a $CMS_{\mathbf{2}}$ will be dealt with in future developments.

ii.) Development boards are provided by a silicon chip manufacturer for fast prototyping. A PCB design embedding the SoC will be redesigned and developed in future work.

iv.) The work conducted in this dissertation contains the preliminary developments of the proposed HTEL anti-theft application with the following limitations:

- Only PCDs running the Android operating system will be considered in this study.
- This study will not focus on ROM (Android application, operating system, and firmware) development to render a PCD inoperable.

1.8 Outline of Dissertation

This dissertation is outlined as follows:

- **Chapter 2** deals with a comprehensive technical overview of low-power wireless technologies with IoT capabilities relevant to the study. The concept-to-design phase of the HTEL anti-theft system is presented and conclusions presented.
- **Chapter 3** deals with the experimental study consisting of the development of a mesh network in an indoor environment.
- **Chapter 4** deals with results and presents the analysis of the data collected in the mesh network experimental study.
- **Chapter 5** discusses the important findings of this research. The research goals reached are outlined and future work presented.

Chapter 2

Low-Power Wireless Technologies

In this chapter, work is presented on the HTEL anti-theft initiative by discussing some of the designed concepts in the preliminary study that may potentially provide solutions to the problem statement. Shortcomings in the application design during the concept phase are identified, and significant improvements are made in the final design that consequently leads to the project implementation phase using state-of-the-art low-power and low-throughput wireless technologies. The final proposed network design of the anti-theft system with a technical overview of the wireless technologies is presented and the network developed, which as will be shown, could be used for almost any type of instrumentation application that requires low-power wireless communications. A detailed description of the protocols are given, and their implementation uses a high-level Application Programming Interface (API) on a System on Chip capable of running multiple wireless protocol stacks. The SoC referred to in this study is an integrated 32-bit MCU and 2.4 GHz radio embedded on a silicon chip. The API provides functions to access the layers of the wireless technology

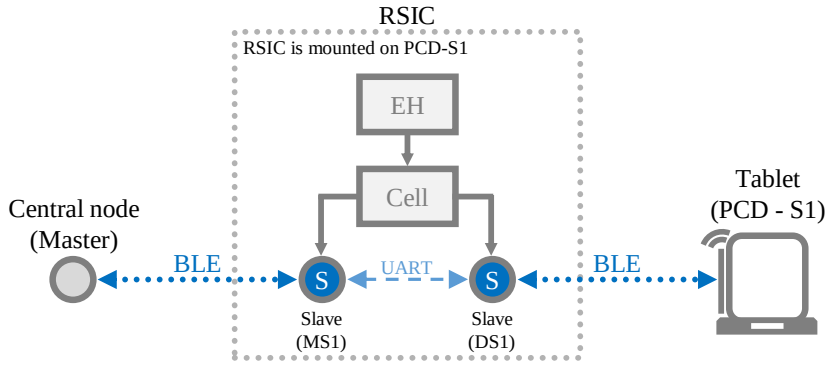


Figure 2.1: Preliminary design concept using an RSIC mounted on the PCD and communicating via BLE to a central node and tablet [20, p.2]. An energy harvester denoted by EH, provides charging capabilities to a battery cell that provides power to the slave nodes.

protocol stack. A particular emphasis is placed on the hardware layer of the wireless technologies applicable to the research with a particular focus on the HTEL anti-theft solution in protecting educational devices. The systematic approach followed in the study of low-power wireless technologies results in the proposition of an IoT solution that operates on the IEEE 802.15.4 [19] hardware layer of the Open System Interconnectivity (OSI) model used in telecommunications. Conclusions are presented at the end of the chapter that leads to the development of a mesh network conducted in the experimental study.

2.1 Concept-to-implementation phase

During the inception phase of the anti-theft initiative, a concept was derived with the following design requirements articulated from the HTEL. With reference to **Figure 2.1**, the concept design includes:

1.) Incorporating a miniature embedded platform referred to in this study as a Radio-Slave Interface Circuit (RSIC) mounted on the Portable Computer Device (PCD). An RSIC is a multi-layer PCB design with an area not exceeding 25 cm^2 consisting of two slave SoCs running BLE, an energy harvesting IC charging a rechargeable pocket-sized battery, active-passive components and a programming input.

2.) Supply power to the SoCs of the RSIC using a coin cell battery and integrated with an energy harvesting IC that harvest energy from the environment.

3.) Incorporating a low-power wireless technology between the device and network node that is compatible with devices running on Android operating systems, such as BLE.

The design concept illustrated in **Figure 2.1** was presented by van Rensburg [20] as a proof of concept at the High Energy Particle Physics (HEPP) workshop based on the requirements layout by the HTEL. At the time, little was known about the accessibility and availability of low-power wireless technologies on Android devices or whether a proprietary wireless protocol had to be developed for the HTEL anti-theft system. It was realised during the study that smart-devices running iOS and Android operating systems incorporate BLE and that the technology may be used to communicate to intelligent embedded platforms capable of running multiple wireless technologies concurrently on a single SoC. BLE has shown significant growth in all types of smartphones and tablets. The technology is currently evolving and fast becoming the preferred technology for low-power applications. Major technology companies and silicon chip manufacturers invest a substantial amount of effort in research and development of the BLE protocol stack [21]. For this reason, the application mainly focusses on BLE for communications from a PCD to an external node in the mesh network. Furthermore, the design concept to render a device inoperable using an

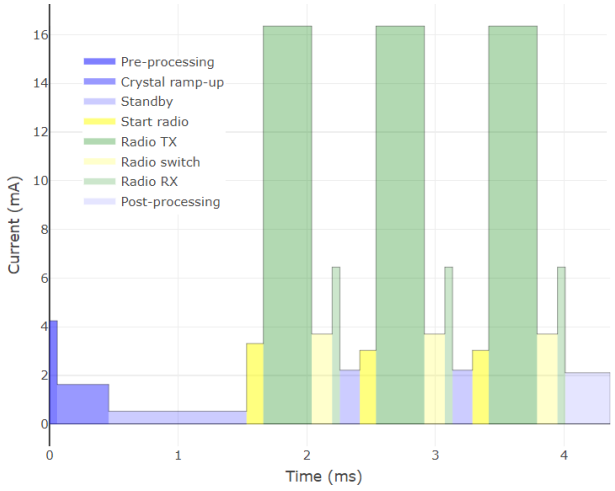
Table 2.1: Parameters and settings obtained from a) Online Power Profiler [24] estimating the lowest b) average power consumption of a slave's advertisement over a 10245 ms interval with a BLE event of 4.36 ms.

(a) nRF52840-SoC test-setup and BLE information.

Parameter	Value	Description
Role	Advertiser	Advertisement to master
Voltage	3.0 V	SoC supply voltage
Softdevice	s140	Protocol stack software
Interval	10245 ms	Advertising interval
Length	4.36 ms	BLE event length
Tx power	8 dBm	Radio transmitting power
Tx payload	31 Bytes	Transmitted packet size

(b) nRF52840-SoC current consumption per BLE event.

Parameter	Value	Description
Charge	25.46 uC	BLE event charge
Idle current	2.7 uA	BLE event idle current
Average current	5.2 uA	BLE event average current



RSIC proposed in van Rensburg [20, p.3], also introduced a star topology where a master may connect multiple slave RSICs simultaneously. A slave is a node that will advertise its presence in the building space to a master. A master is a node that connects and initiates the transmission of data to a slave node. Consequently, a design was put in place with two SoCs that both run BLE and powered from a coin cell (CR2032) battery. The battery is charged from an energy harvesting IC with highly optimised DC_1-DC_1 boost converters and power management circuitry capabilities such as the LTC 3108 [22] or MAX 17710 [23] using either RF or thermal sources or both from the environment.

With reference to **Table 2.1**, it was estimated from the Online Power Profiler [24] that an nRF52840-SoC configured as a slave node and transmitting at 8 dBm (6.31 mW) transmitting power with a 31-byte payload, would only consume an average of 15 uA current over ten-second regulatory updates. The Battery Life [25] expectancy = $(300\ \text{mAh} \times 3\ \text{V})(5.2\ \text{uA} \times 3\ \text{V})^{-1} \times 70\% =$ 40385 hours and reveals that a slave node will run for more than four years, without charging, on a coin cell rated at 300 mAh at an efficiency of 70% [20, p.6].

The study in van Rensburg [20] further proposes algorithms with reference to **Figure 2.1** where BLE is used for communications between the device (PCD - S1) to slave (DS1) and the central node (master) to slave (MS1), to render a device inoperable. Both slaves, DS1 and MS1, communicate through a serial interface to exchange packets between the two ends while communications between the device (PCD - S1) to slave (DS1) and the central node (master) to slave (MS1) is wireless. It is important to note that BLE is connection orientated and therefore the slaves (DS1 and MS1) will advertise their connections to the masters (PCD and central node) while the master nodes will be responsible for

connecting to the slaves and initiate the data transfer. As the description above entails, the incorporation of an RSIC required that both the PCD and a central node be configured as master nodes. The RSIC design using the power consumption specification in **Figure 2.1** also required a comprehensive study into the feasibility of energy harvesting using RF or thermal sources to power the two slave nodes.

At the time, this was only a concept that lead to the realisation of a significant flaw in the HTEL design specification by using the RSIC mounted on the PCD. The RSIC added much unneeded complexity to the application in terms of communications and to render a device inoperable. A better alternative is the omission of the RSIC in the design and to configuring the PCD as a slave node for direct communications to the central node. However, the star network itself had some pitfalls by covering only a small space in a building area. More extensive network coverage would require multiple central nodes without being able to communicate to a $CMS_{\mathbf{2}}$. It was later discovered in the literature that multi-protocol SoCs such as the nRF52[1] from Nordic Semiconductor could run BLE and IEEE 802.15.4 [19] mesh wireless technologies concurrently and the need for any external electronics mounted on the device exhibits an unnecessary complexity in the design.

In summary, the evolution of the technological development of the design has shown the following progress during the inception phase of the anti-theft initiative:

1.) Concept phase design, part I [20, p.2]: The solution consists of a star topology where multiple devices may be connected to a master node. More PCDs may be connected by disconnecting already connected devices. A major shortfall in the concept is the RSIC mounted on the PCD. The star topology using BLE is unable to cover large

[1] www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840.

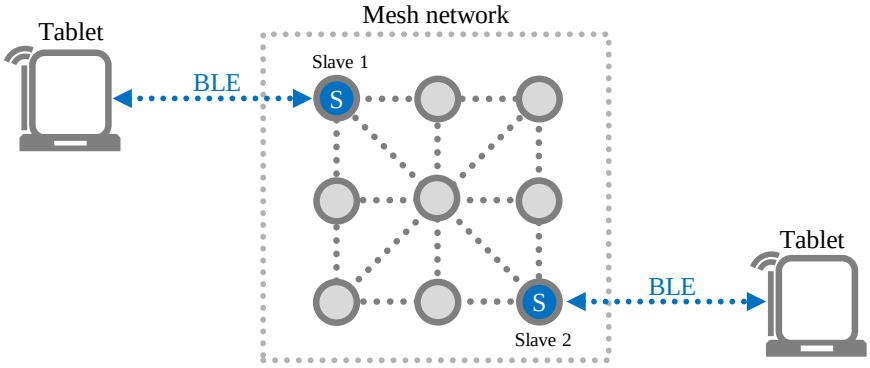


Figure 2.2: Final design concept omitting the RSIC mounted on the PCD and communicating via BLE to the nearest node in a mesh network.

areas and communicating to a CMS_2.

2.) Concept phase design, part II [20, p.4]. Included a mesh network design concept that is able to communicate to a CMS_2. The design did not yet fully disregard the concept of an RSIC mounted on the PCD or embedded in the PCD PCB.

The implementation phase consisted of a final design taking into account the shortcomings of the concepts discussed above. This lead to an implementation phase involving the development and analysis of a mesh network in the research conducted by van Rensburg et al. [26]. As depicted in **Figure 2.2**, the implementation phase omits the custom RSIC design shown in **Figure 2.1**, whereby a PCD directly communicates via BLE to the nearest node in an IEEE 802.15.4 [19] build mesh network. A node in the network represents a multi-protocol SoC where multi-protocol means running two different wireless technologies at the same time on a single SoC, for example, BLE and Thread or BLE and ZigBee or BLE and an IEEE 802.15.4 [19] proprietary. In the upcoming sections of the chapter, a comprehensive overview of a low-power star and mesh wireless technology relevant to this study is presented, with a particular focus on the HTEL anti-theft application.

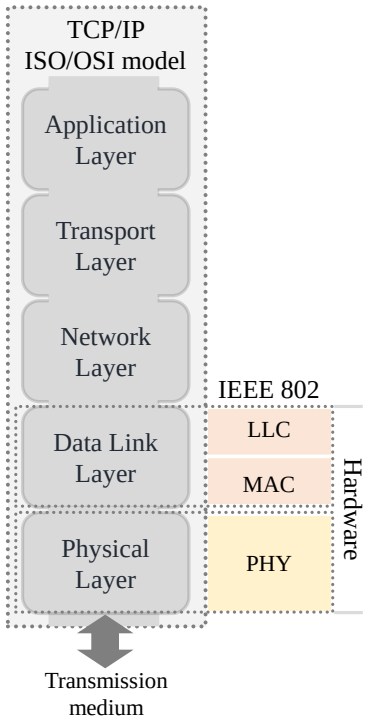


Figure 2.3: Protocol stack of TCP/IP communication system based on the OSI model illustrating the IEEE 802 hardware layer.

2.2 Low-Power Wireless Technologies

In this section, we briefly discuss the layers of the Transmission Control Protocol - Internet Protocol (TCP/IP) protocol suite based on the OSI model. The introduction will lead to an in-depth illustration of two IEEE 802.15 standards for communications of low-power Wireless Personal Area Networks (WPANs). WPANs are defined as independent ad-hoc networks that transmit and receive information between nodes over limited distances. Ad-hoc in this study means autonomous decentralised wireless networks that do not necessarily rely on other network infrastructures to function. These type of network systems allow cost-effective and power-efficient solutions to wireless networking between

multiple interconnected devices in a wide range of commercial and industrial applications [27, p.1]. Electrical engineering societies and coordinating committees developed the IEEE 802.15.4 [19] standard that mostly focuses on the hardware layer of the OSI model.

As mentioned in **section 1.5**, this study mainly involves an experimental study by the development of a low-power mesh network using an IEEE 802.15.4 [19] wireless technology. The wireless technology incorporate the two sub-layers of the hardware layer of the OSI model. For this reason, it is crucial to understand how the different protocols that compose of a particular wireless technology is integrated into the OSI model, with a special interest in a low-power IP based wireless technologies. The understanding of these protocols will further aid in the analysis of potential problems in terms of performance and reliability of the developed mesh network or in the development of new protocols. The OSI model that makes up these protocol layers is stacked in a hierarchical structure shown in **Figure 2.3**. Each bottom layer is controlled by protocols to communicate with the next higher layer using services. The higher layers request services from the lower layers. In summary, the layers of the OSI model relevant to this study are briefly defined from bottom-up as follows [28, P.9-10]:

The Physical Layer (PHY) uses a transmission medium to send and receive electromagnetic signals between devices. The layer is responsible for modulation (digital to analog transmission) and demodulation (analog to digital transmission) of the data.

The Data Link Layer (DLL) comprises upper Logical Link Control (LLC) and lower Media Access Control (MAC) sublayers. LLC is responsible for framing, flow and error control of the transmission data when communicating between two nodes. Flow control is the sending frames from a source to the destination node in such a manner that the

receiving node process the received packets at the same rate that the packets are delivered. A frame of characters encapsulates a flag at the start of the frame, a header containing the source and destination address, an 8-bit message (payload), trailer for error detection and other control information and a flag consisting of a an 8-bit character at the end of the frame [29, p.294-295]. Decapsulation occurs at the receiving node lower layer. If the frame is intended for the receiving device, the frame is stripped of its headers at the LLC before being passed to the network layer. If the demand is higher or lower than the supply, the receiving node provides flow control feedback to the transmitting node to stop or throttle down transmitted frames. Buffers with memory locations can be used at both sending and receiving nodes to hold packets [29, p.297]. The digital transmitted signal at the source over the transmission medium is checked for errors at the destination node. The electromagnetic signal over the air may be susceptible to errors. If an error is detected at the receiving node, it needs to be corrected or re-transmitted [29, p.241]. MAC layer is responsible for multiplexing and demultiplexing and multiple access with or without collision avoidance mechanisms. Multiplexing occurs at a source node where higher protocol layers encapsulate a packet to lower layers while demultiplexing occurs at a destination where lower protocols decapsulate a packet to higher layers.

The Network Layer is responsible for managing connections in the network by providing services for addressing and routing of data between nodes in the network. For IP based networks, each router in the network connected to neighbouring routers share link addresses and link costs to optimise the routing of a packet. The Internet Control Message Protocol (ICMP) is one of the protocols incorporated at the network layers to report problems when sending a packet from source to destination.

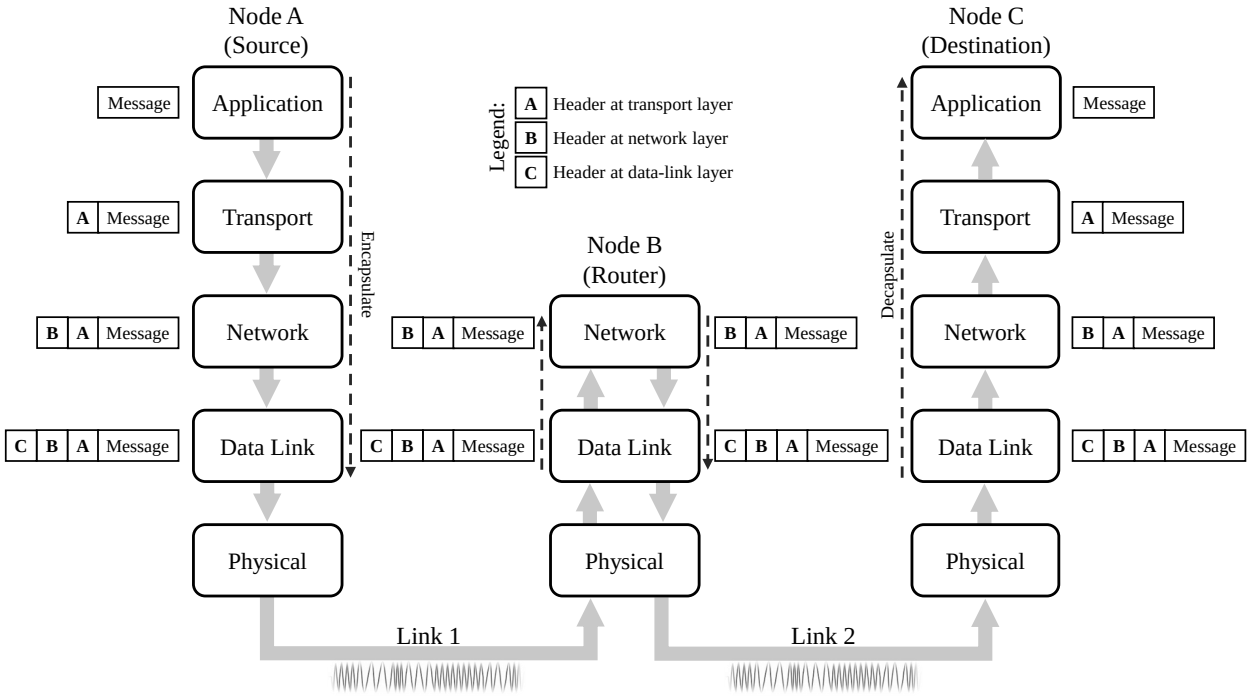


Figure 2.4: TCP/IP protocol suite adapted from Forouzan [29, p.36, 41] with the illustration of multi-hopping packet transmission over a two-hop distance using an intermediary router over two links. A packet is encapsulated from the source node and decapsulated at the receiving node.

The Transport Layer runs protocols such as the TCP and UDP. TCP is a connection oriented protocol to send packets using the port and IP addresses that use acknowledgements to indicate successful delivery of a packet. UDP on the other hand is a connectionless oriented protocol to to send packets using the port and IP addresses that do not use an acknowledgement to indicate successful delivery of a packet.

The Application Layer is responsible for the messaging between low-power devices that run protocols such as the COnstraint Application Protocol (COAP).[2] COAP is a specialised protocol on the application layer used for constraint devices in WSNs to communicate over the internet. Each router distributed in the environment will consist of an application layer to send and receive messages through the mesh to the CMS_2 or cloud service. Depending on the application, typical information may include the device identifier, sensor measurements or router location in the environment where a PCD or sensors are connected.

A packet is sent and received through the TCP/IP protocol layers between two devices over the transmission medium. As illustrated by **Figure 2.4**, a message called a packet is produced from the application layer without containing any headers or trailers. The transport layer reads the packet which is called the segment in TCP and the user datagram in UDP and encapsulates the message with a header. The header at the transport layer may contain information for flow and error control. The transport layer then passes the packet now known as a datagram to the network layer for further processing.

The transport layer then passes the packet now known as a datagram to the network layer for further processing. The network layer further encapsulates a header to the data-

[2]Constraint is referred to low-power and low-throughput devices with limited processing and memory capabilities.

gram packet containing the source and destination address. The data link layer takes the datagram packet from the network layer and encapsulates a header containing information of the next-hop addresses to a frame. Finally, the frame is then transmitted from the physical layer over a transmission medium. When the data is received at the router, the router uses the PHY, DLL and network layer to relay the frame to reach the destination node. The DDL of the next node (B) decapsulates the frame to a datagram before passing the datagram to the network layer. The network layer protocol then inspects the source and destination address and consults a routing table using an algorithm in order to relay the datagram to the next node. Node's B data link layer then encapsulates the datagram in a frame and passes it to the physical layer for transmission over the medium to the node C. The reversed process is known as decapsulation when the frame arrives at the destination node [29, p.42]. Error checking is conducted at the destination node, and an acknowledgement is transmitted to the source node to indicate whether the packet was successfully delivered.

We recall that the layers of the OSI model consist of protocols with services. The protocols are a set of rules for data communications between layers and should not be confused with algorithms which deal with producing some output or outputs with single or multiple inputs. One may say that algorithms exist in a layer that computes a particular communication technology based on the inputs received and passes the information when requested by the next upper layer. Some technologies perform better than others but ultimately produce identical outputs when receiving the same inputs. A service routine provides functions to the higher layers of the protocol stack, and the protocols execute the tasks defined by the service routine [28, p.13]. The protocols were described in **Figure 2.4**

to illustrate how a message passes through the layers. Protocol layering takes the complex task of communicating between layers and divides them into smaller more straightforward tasks. Different wireless technologies may use layers with different algorithms and therefore some algorithms may outperform others depending on the technology used.

In conclusion, the TCP/IP protocol suite in **Figure 2.3** is used as a model that aids in the understanding and development of the protocol layers that consists of different wireless technologies with a particular focus on the IEEE 802.15 hardware layer. There are subtle differences in the communications between protocol layers and different algorithms being used. However, the working principle of the PHY and DLL layers of these wireless technologies are almost identical and discussed in the upcoming sections.

2.3 Bluetooth low-energy

Jaap Haartsen first invented Bluetooth under the mobile phone division of Ericsson in 1994. The technology was later adopted by the Bluetooth Special Interest Group (SIG) with the goal of developing WPAN technologies and is currently managed today by more than 30,000 member companies in the telecommunications industry. In 2002, the IEEE approved the Bluetooth specification layout from the Bluetooth SIG known today as the IEEE 802.15.1 [27] standard for WPAN networks. However, the technology is no longer maintained by the IEEE but by companies that are members of the Bluetooth SIG. Technology manufacturers that plan to market the Bluetooth technology in their products must meet the Bluetooth SIG standard. SIG membership does not pertain to users who develop Bluetooth applications on computer devices to communicate to external devices. Universi-

ties or companies can join for free as 'Adopter' memberships that entitle developers to have access to the published specifications to build their products using Bluetooth technology.[3]

Bluetooth mainly consists of the original standard specified by the Basic Rate (BR) and the current Bluetooth Low-energy (BLE) version first introduced in 2010 [28, p.51]. The BR version dealt with the core versions 1.0 to 3.0 while BLE was first introduced in version 4.0 where the SIG developed a new protocol stack for low-energy applications in 2010.[4] BR was later changed to Enhanced Data Rate (EDR) by the modification of the MAC and PHY layers for improved data rates. The BR hardware layer uses Gaussian Frequency Shift Keying (GFSK) as modulation technique and is illustrated by the mathematical equation in Burbank et al. [28, p.53]. The GFSK modulation method has shown to be more bandwidth efficient than FSK alone in terms of data rates while minimising intersymbol interference resulting in better reliability. In the EDR system, the modulation method is changed for each packet send using GFSK for sending a packet header at 1 Mbps and Differentially-encoded Quadrature Phase Shift Keying (DQPSK) and Differentially-encoded Phase Shift Keying (DPSK) for sending data at higher 2 Mbps and 3 Mbps rates, respectively.

Data throughput between communicating nodes is actually significantly lower because of bidirectional traffic, protocol overhead, MCU/CPU and radio limitations [30, p.7]. To illustrate throughput overhead, a particular low-cost SoC can transmit 8 data packets per connection interval and each data packet can obtain 18 bytes of user data transmitted per connection interval. Assuming the shortest connection interval is 10 ms (data transmissions between two peers per connection interval), this provides a maximum of 1/10 ms = 100

[3] www.bluetooth.com/develop-with-bluetooth/join/.

[4] BR/EDR versions 1.x to 3.x are not directly compatible with the BLE versions 4.x to 5.1. The upper layer protocols are different between versions [30, p.3].

connection events per second and 144 bytes per connection event (8 packets × 18 user bytes per packet) that is continually transmitting at the maximum data rate of the SoC. This will result in a throughput of 100 connections events × 144 bytes = 14.4 kBps or 0.12 Mbps, which is significantly lower than previously stated. At the minimum advertising interval and maximum bytes per connection rate results in the SoC transmitting at maximum power consumption. To avoid this, the advertising interval increased as illustrated in **Table 2.1** resulting in the radio being switched off for more extended periods and saving considerate battery power [30, p.8].

Specific channels exist in the ISM band spaced 1 MHz apart and uses the Frequency Hopping Spread Spectrum (FHSS) scheme of different channels to avoid interference problems from other devices operating at the same frequency band. Bluetooth hops between the 79 different channels at 1600 times per second on the 2.4 GHz ISM band to transmit data between devices. A device in the network can therefore spend 1/1600 s before it moves on to the next frequency to communicate to other devices in the network [29, p.457].

In BLE, major improvements were made from the BR/EDR on the Physical and link-layer resulting in improved energy efficiency in Bluetooth devices [28, p.66]. BLE was specifically designed for low-power and low-throughput wireless communications between devices over short distances. The wireless technology uses Gaussian Minimum Shift Keying (GMSK) modulation that results in higher and reliable transmission ranges over BR/EDR versions using the same output power ranges. A channelisation technique is used that operate on the 2.4 GHz ISM band consisting out of 40 channels. A channel is a frequency on which a signal is modulated and demodulated, starting from channel 0 at 2402 MHz to channel 39 at 2480 MHz spaced 2 MHz apart. Only three channels are used for advertising

and 37 channels are dedicated for data channels. Like BR/EDR, BLE also incorporates frequency hopping on the PHY layer that uses unique address identifiers at the beginning of each packet on an RF channel to transfer data. The use of FHSS carriers on the ISM band-aid modulated signals against interference from other transceivers operating at the same transmitting frequency [31, p.183].

BLE uses five states for communication between devices and only one state can be active at a time [28, p.67]:

- 'Standby state': A device is unable to send and receive packets.
- 'Advertising state': A device transmits advertising channel packets and listen for responses.
- 'Scanning state': A device will listen for advertising channel packets.
- 'Initiating state': A device will listen for advertising channel packets from a particular device before initiating a connection with that device.
- 'Connection state': A device connection is maintained with another device and data is transferred over the data channels.

The network topology of BLE is the same as the BR/EDR system and consists of master and slave nodes in a network. A Bluetooth network is called a piconet consisting of a single master node and multiple slave nodes forming a star topology. A limited number of slave nodes can connect to the master node while slave nodes are not allowed to connect to each other. Piconets can be combined to form scatternets [29, p.452]. BLE is not initially designed for high-throughput applications such as voice or data transmissions but rather low-power consumption sensor applications for monitoring and control or to communicate to other BLE capable devices.

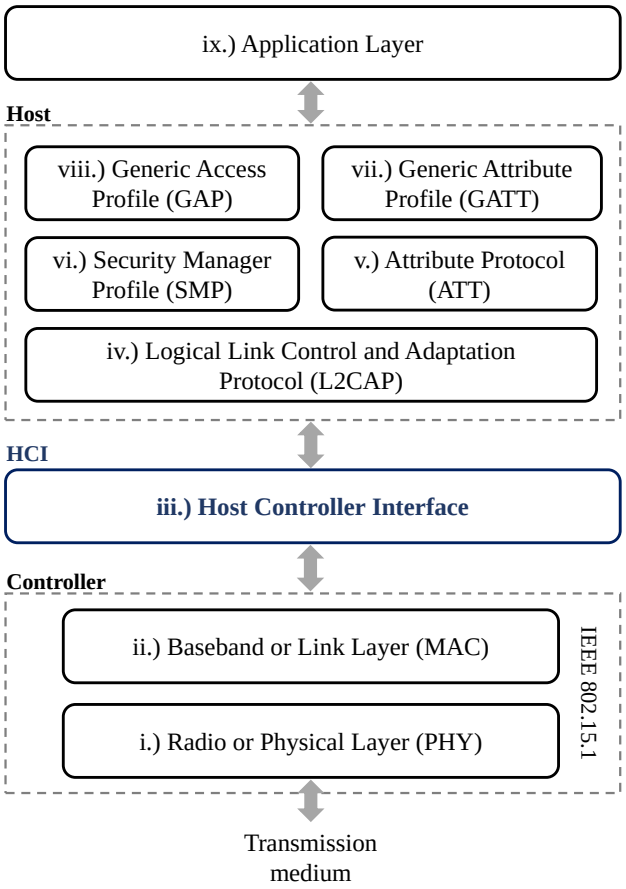


Figure 2.5: BLE protocol stack adapted from Townsend et al. [30, p.16].

A description of a network topology relevant to this research for device communications to a mesh network is discussed in **section 2.3.3**. We will discuss in detail the BLE hardware layer to understand how the layers can be translated into the OSI model. With relevance to the anti-theft initiative, BLE is incorporated in almost all smart-devices today and the preferred wireless technology for device communications to the mesh network.

2.3.1 IEEE 802.15.1

The IEEE 802.15.1 [27] standard was initially formulated for the classic BR/EDR versions that focus on the MAC and PHY specifications of the Bluetooth hardware layer. BLE, also known as Bluetooth Smart which can implement both BR/EDR and BLE versions, builds from the standard in the Bluetooth 4.0 core specification and is now maintained by the Bluetooth SIG. The standard defines the radio and baseband layer that is equivalent to the PHY and MAC layer of the OSI model [29, p.452, 457]. The architecture of the BLE protocol stack is discussed in the next section.

2.3.2 Protocol stack

Major smartphone manufacturers invest significant resources in producing a reliable protocol stack for the BLE wireless technology. Consequently, with the amount of research and development conducted in BLE, many silicon chip manufacturers have adopted the technology in their products and provide BLE solutions to developers. Chip manufacturers provide low-level APIs in order to use the BLE framework or protocol stack to software developers for their applications [30, p.2]. The latest BLE core specification, version 5.1[5], was released in 2019 and includes a full technical description of all the technology building blocks regarding the protocol stack. Some of the new features in this version is accurate direction identification of transmitting Bluetooth devices from a receiving antenna that may greatly aid in indoor localisation techniques in a building for tracking purposes [31, p.281-284]. As depicted in **Figure 2.5**, the main three building blocks of any Bluetooth

[5] www.bluetooth.com/specifications/bluetooth-core-specification/.

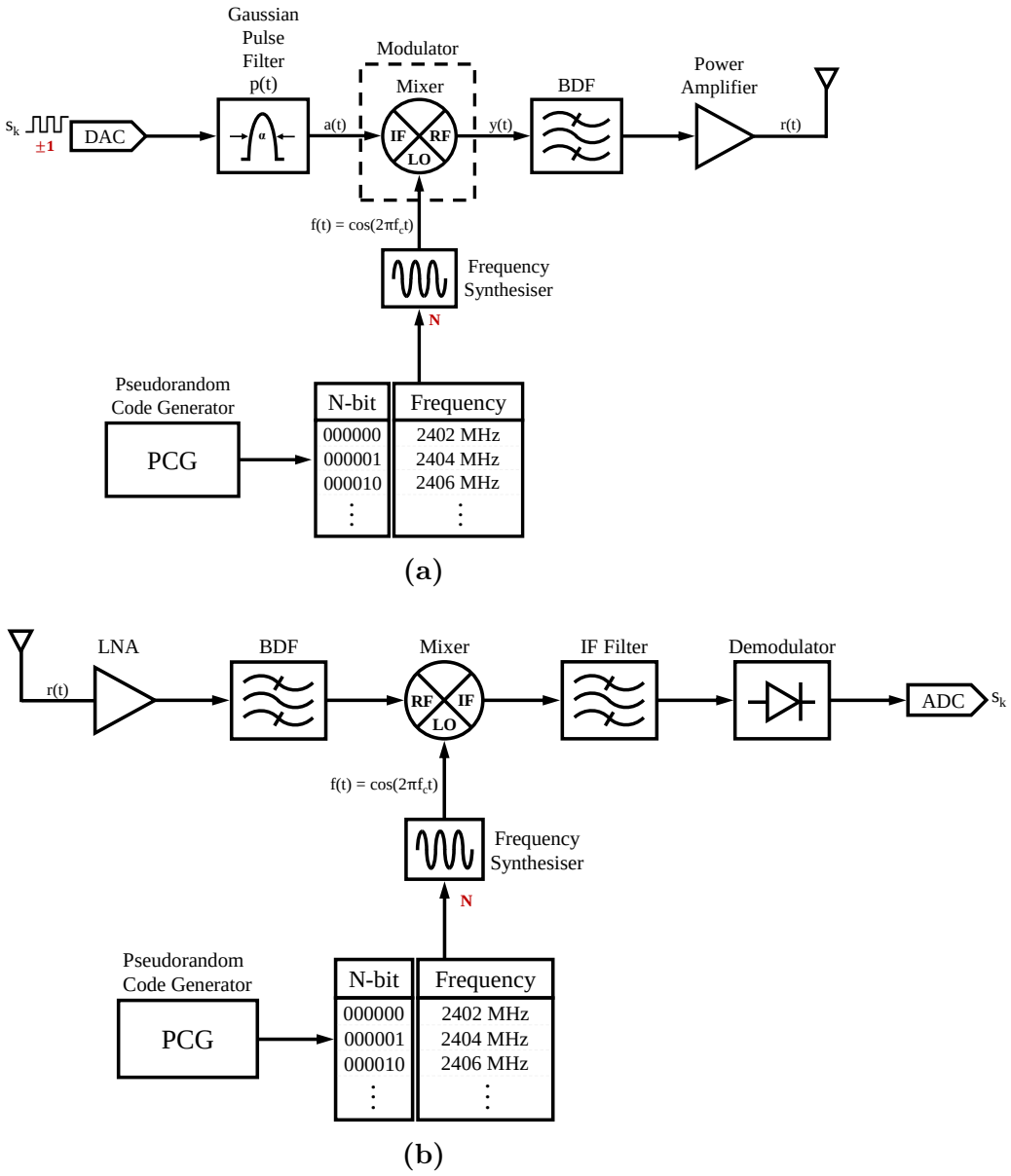


Figure 2.6: Conceptual heterodyne a) modulation of binary data s[k] to generate r(t) and b) demodulation of analog signal r(t) to generate s[k].

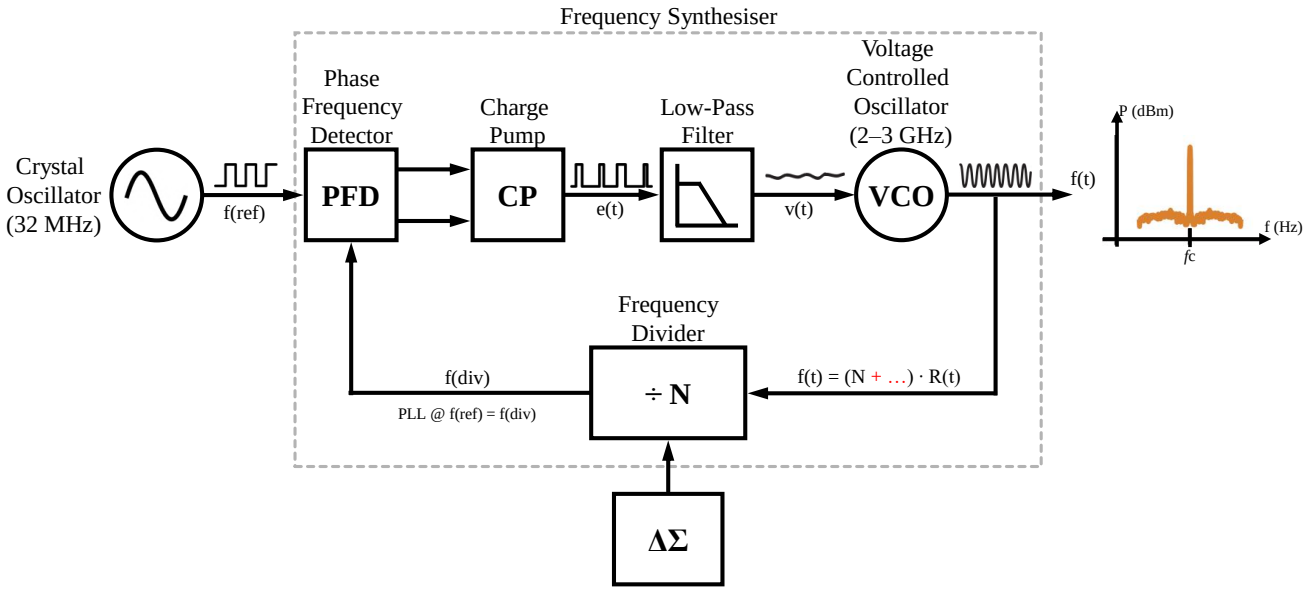


Figure 2.7: Closed-loop fractional-N phase PLL frequency synthesiser control system with $\Delta\Sigma$ modulation modified from Lacaita et al. [33, p.14-72] and Su et al. [34, p.881].

device includes the application, host and the controller. The application contains a user-developed program that interface with the Bluetooth protocol stack usually through the silicon chip manufacturer's API.

The host consists of the upper layers while the controller consists of the lower layers (based on the IEEE 802.15.1 [27] standard MAC and PHY specifications) of the Bluetooth protocol stack. In the specification, there also exists a communication protocol between the host and controller known as the Host Controller Interface (HCI). The HCI's main objective is to interface the host and controller layers in the protocol stack produced by different software companies. Silicon chip manufacturers such as Nordic Semiconductor and Texas Instruments integrate the application, host and controller systems on a single SoC resulting in the reduction of hardware development and leading to a low complexity in the initial PCB design.[6]

Compared to other wireless technologies that provide higher data throughput, such as WIFI and classic Bluetooth, the BLE protocol stack is designed for low-energy and low-data throughput wireless applications. This is achieved programmatically by advertising packets at a lower Duty Cycle (DC_2) through the API. The main feature of BLE is that it is a connection oriented technology and the device to be connected will advertise packets to nearby devices containing a 31-byte payload with the transmitter address information [30, p.10]. The DC_2 in terms of battery consumption refers to the on-time (when transmitting a packet) over the advertising interval time during a connection event. For example (with reference to **Table 2.1**), assuming a 31-byte advertising packet takes 102.4 ms to transmit

[6]Hardware platforms running BLE requires at least a 16 MHz ARM Cortex-M0 core MCU with 256 kB of flash and 16 kB of SRAM memory to run the protocol stack on a single SoC.

to a device to be connected and the connection interval is 10240 ms, will result in a $DC_2 =$ (102.4 ms ÷ 10240 ms) ×100% = 1% [20, p.6], resulting in saving significant battery power of the transmitting device or tablet. A smaller data packet will take less time to transmit per the advertising connection interval and will cause a lower $\mathbf{DC_2}$ saving a significant amount of battery power.[7]

With reference to **Figure 2.5**, the main building blocks of the BLE protocol stack are the 'controller' and 'host'. The 'controller' consist of the following layers:

i.) Physical Layer (PHY):

Mainly responsible for transmitting and receiving electromagnetic signals for radio communications over the air and the processing of these signals in a usable digital data format by the MCU. The IEEE 802.15.1 [27] initially outlined the PHY specification and is currently maintained by the Bluetooth Core Spec [31]. The radio of the SoC operate on the unlicensed 2.4 GHz ISM frequency band and no legal requirement is needed to buy a license and operate devices on the ISM band frequency for wireless communications, provided transmitters do not exceed 10 dBm transmitting power. As illustrated in **Figure 2.6**, BLE uses GFSK modulation of the digital baseband signal s[k] to an anolog r(t) signal for transmission over the air. An analog electronics circuit or software-defined radio is used to modulate or demodulate a signal known as a mixer. A mixer inputs the binary signal s[k] denoted as the Intermediate Frequency (IF) and the carrier signal f(t) denoted as the Local Oscillator (LO) to modulate an RF output signal r(t). The carrier is a periodic frequency operating at the 2.4 GHz ISM band consisting of 40 channels with 2 MHz spacing. Demodulation, in essence,

[7]The nRF52840-SoC provides data transfers of advertising connection intervals between 20-10240 ms.

is the reverse process of modulation of the baseband signal r(t) to a binary signal s[k]. Before modulation, it is important to note that a rectangular data pulse from the MCU in the time domain will produce an undesirable range of energies and therefore an infinite bandwidth in the frequency domain (one main-lobe and side-lobes on each side of the main-lobe) which will cause interference on other channels. A Gaussian Pulse Filter (GPF) is used before FSK modulation to prevent spreading (limits the bandwidth of the transmitted pulse) of higher and lower frequencies represented by these side lobe energies causing interference. With reference to the heterodyne GFSK modulation block diagram illustrated in **Figure 2.6a**, the continuous-time pulse shaped function $p(t)$ is given by:

$$p(t) = \begin{cases} B_w T_s \sqrt{\frac{\pi}{\alpha}} \exp\left[\frac{-\pi^2 t^2}{\alpha^2}\right], \alpha = \frac{1}{B_w T_s}\sqrt{\frac{log\, 2}{2}} & , 0 \leq t \leq T_s \\ 0 & , \text{otherwise,} \end{cases} \tag{2.1}$$

where B_w is the 3 dB bandwidth and T_s is the bit period or baseband symbol duration in seconds. The roll-off factor α is related to the 3-dB bandwidth. The bandwidth bit period is given as $B_w T_s$ in which the symbol period T_s shall be 1×10^{-6} seconds [31, p.356].[8] A smaller B_w leads to a wider spectrum but will be more susceptible to Inter Symbol Interference (ISI). ISI causes distortion of the baseband signal when symbols interfere with subsequent symbols. According to the Bluetooth Core Spec [31, p.352-353], the GPF use a $B_w T_s$ of 0.5 and the modulation index shall fall between 0.28 and 0.35. Note that for a $B_w T_s$ of 0.5, the tails of the filter is spread over two-bit

[8] Ignoring packet overhead, a character containing 8 bits of data will only take 8 us to sent by the radio.

periods. The continuous-time pulse shaped baseband signal is given as:

$$a(t) = \sum_{k=-\infty}^{\infty} s_k p(t - kT_s), \tag{2.2}$$

this results in the mixer (modulator) producing a Continuous Phase Frequency Shift Keying (CPFSK) output $y(t)$ defined as [32, p.103,104]:

$$\begin{gathered} y(t) = \sqrt{\frac{2E}{T_s}} \cos\left[2\pi f_c t + \frac{\pi h}{T_s} r_k(t - kT_s) + \pi h \sum_{i=1}^{K} r_i\right], \\ kT_s \leq t \leq (k+1)T_s, \end{gathered} \tag{2.3}$$

where E is the energy per symbol for the modulation, T_s is the bit period in seconds, f_c is the carrier frequency in hertz. Based on the NRZ binary coding scheme, $r_k = \pm 1$ is the k^{th} data bit representing a logic one (positive voltage) and zero (negative voltage) with pulse shaping not included and h is the modulation index. The second term in **Equation 2.3** represents the information-carrying phase, while the third term represents the accumulated phase. The modulation index h is defined as:

$$h = \zeta_f \times T_s,\ \zeta_f = 2 \times \Delta f,\ T_s = \frac{1}{S_r} \tag{2.4}$$

where ζ_f is the total frequency deviation from the carrier frequency f_c that represents two symbols (bit one and bit zero for binary FSK), S_r is the symbol rate at 1 Mbps

and T_s is the bit symbol period at 1×10^{-6} s. Furthermore:

$$\zeta_f = f_1 - f_0,\ f_1 = |f_c + \Delta f|,\ f_0 = |f_c - \Delta f|, \tag{2.5}$$

where ζ_f is the absolute difference between frequency f_1 representing a bit one and frequency f_0 representing a bit zero. Δf is constrained to 140 kHz $\leq \Delta f \leq$ 175 kHz to meet the BLE modulation index specification. For example, to calculate the modulation index h for a given carrier frequency $f_c = 2408$ MHz, let
$f_0 = 2408$ MHz-150 kHz $= 2407.85$ MHz, $f_1 = 2408$ MHz$+150$ kHz $= 2408.15$ MHz
$\zeta_f = (2408.15 - 2407.85)$ MHz $= 300$ kHz, results in $h = 300$ kHz $\times 1 \times 10^{-6}$ s $= 0.3$.
To generate a range of carrier signals consisting out of 40 channels, user data packets are transmitted using channels in the range [0,36] and advertising data packets are transmitted in channels 37, 38, and 39 given by:

$$f_c = (2402 + 2k) \text{ MHz},\ k = 0, 1, 2, \ldots, 39, \tag{2.6}$$

where f_c is constrained to 2402 MHz $\leq f_c \leq$ 2482 MHz. A Frequency Synthesiser (FS), depicted in **Figure 2.7** is a closed-loop feedback digital control system and integrated circuit design embedded in the radio of the SoC. The FS is responsible for generating the carrier signal f_c in **Equation 2.6** using a pseudorandom code generator and frequency table. The Voltage Controlled Oscillator (VCO) provides

high-frequency oscillations but is inherently unstable due to noise and temperature fluctuations. The 32 MHz crystal oscillator is, however, a very stable active or passive device but does not provide the high-frequency oscillations of the VCO. An error signal is generated by comparing $f_{(div)}$ with the input reference $f_{(ref)}$ that is fed to a charge pump circuit and a low-pass filter to the DC_1 voltage controlled VCO. A frequency divider is used to control the desired carrier frequency f_c through the negative feedback loop. A channelisation technique known as FHSS through the FS is used. The generation of the random carrier frequencies is synchronised with the receiver end which uses the spreading code to demodulate the transmitted signal. A delta-sigma ($\Delta\Sigma$) fractional-N Phase Locked Loop (PLL) is the most commonly used FS used in low-power RF designs. The $\Delta\Sigma$ modulation (a discussion beyond the scope of this literature review) is a complex digital control system design to reduce reference and fractional spurs as well as phase noise in the output carrier frequency generated from the synthesiser. The $\Delta\Sigma$ ultimately reduce ISI arising from multi-path signal reflections. FHSS hops between 40 different frequency channels and therefore prevents eavesdropping and also jamming of the transmitted data over the air. Finally, BLE also incorporates the Time Division Multiple Access (TDMA) protocol for multiple devices to share the same channel. TDMA ensures that devices share the bandwidth of the data channels illustrated in **Equation 2.6** in time whereby each device is allocated a time slot in which it can send data.

ii.) Link Layer:

Similar to the MAC layer of the OSI model and responsible for advertising, scanning, creating and maintaining connections. The link layer ensures encryption and

structure of packets as well the data being transmitted between devices is sent, acknowledged and retransmitted if necessary. Up to 296 bytes of data are usually encapsulated in a packet between an 8-bit preamble, 32-bit access address, 8-bit header, 8-bit length in front and 24-bit CRC at the back. An error checking mechanism is used to check if the packet transmitted is received correctly by generating a CRC code at the end of the encapsulated data. An acknowledgement may be sent to the transmitter before passing the packet over to the network layer to indicate the packet was successfully received. The 8-bit preamble allows the receiver to synchronise bit timing with the transmitter; a 32-bit fixed access address is used for advertising packets but the access address are private for data packets; an 8-bit header to describe the contents of the packet; an 8-bit length to describe the payload length and lastly a 24-bit CRC to check for bit errors in the received packet [35, p.29-30]. See MathWorks[9] documentation 'BLE Link Layer Packet Generation and Decoding' for a comprehensive technical overview of the layer.

iii.) Host Controller Interface (HCI):

An interface that allows communication through a Universal Asynchronous Receiver Transmitter (UART) between the host and controller. Commands and data are sent from the host to the controller and events and data from the controller to the host [35, p.31].

iv.) Logical Link Control and Adaptation Protocol (L2CAP):

Protocol responsible for encapsulation of data from the BLE higher layers into the

[9] www.mathworks.com/help/comm/examples/ble-link-layer-packet-generation-and-decoding.html.

standard BLE packet. See MathWorks[10] documentation 'Bluetooth Protocol Stack' and 'BLE L2CAP Frame Generation and Decoding' for a detailed technical overview.

v.) Attribute Protocol (ATT):

Protocol for accessing data on a peer device.

vi.) Security Manager Profile (SMP):

Protocol for authentication and setting up secure connections between devices.

vii.) Generic Attribute Profile (GATT):

A protocol above ATT that defines the types of attributes and how they can be used.

viii.) Generic Access Profile (GAP):

Defines how devices discover, connect and present useful information to users.

ix.) Application Layer:

Enables the user to interact with the stack through APIs which can extract the characteristics, service and profile specifications between device communications [35, p.35]. The application will be the code that utilises the Android Bluetooth APIs to interact with the Bluetooth hardware when implemented on a tablet.

2.3.3 Network topology

BLE is a connection-oriented wireless technology whereby devices connect to exchange data. The connection between two devices provides secure data transmissions in both directions by periodically exchanging data packets over the connection interval. In BLE network terminology, a central or master is a node that listens to a slave node and initiates

[10] www.mathworks.com/help/comm/ug/bluetooth-protocol-stack.html.

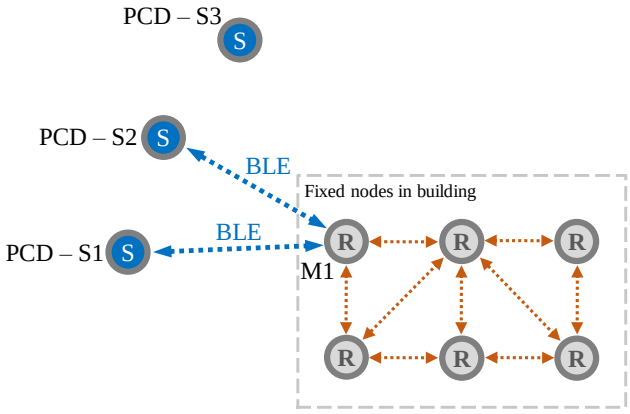


Figure 2.8: Revised BLE network topology proposed for the HTEL anti-theft solution illustrating the connection of multiple slaves nodes (PCD-S1 and PCD-S2) representing tablets to a single master node. Master nodes are configured as routers in an IEEE 802.15.4 [19] based mesh topology.

a connection. The slave is a node[11] that send periodic advertising packets to any available master node for potential connection. As shown in **Figure 2.8**, the BLE network may consist of multiple masters (forming a mesh) where multiple slave nodes can be simultaneously connected to a master node in a star configuration. The link layer only allows a limited number of slave nodes to connect simultaneously to a master node. Once a connection is made to a slave node by the master node, the slave node stops advertising and the master node maintains a periodical data exchange with this, and other slaves connected to it. In the case of the HTEL anti-theft network design, PCDs running BLE will advertise their availability depending on their location to the nearest master nodes in a building area. With reference to the BLE network topology in **Figure 2.8**, an algorithm is proposed in **Figure 2.9** for data communications from a slave node to a master. The proposed algorithm is implemented as an application running on the PCD operating system and is

[11] A node is an SoC embedding an MCU and 2.4 GHz radio capable of running multiple protocol stacks.

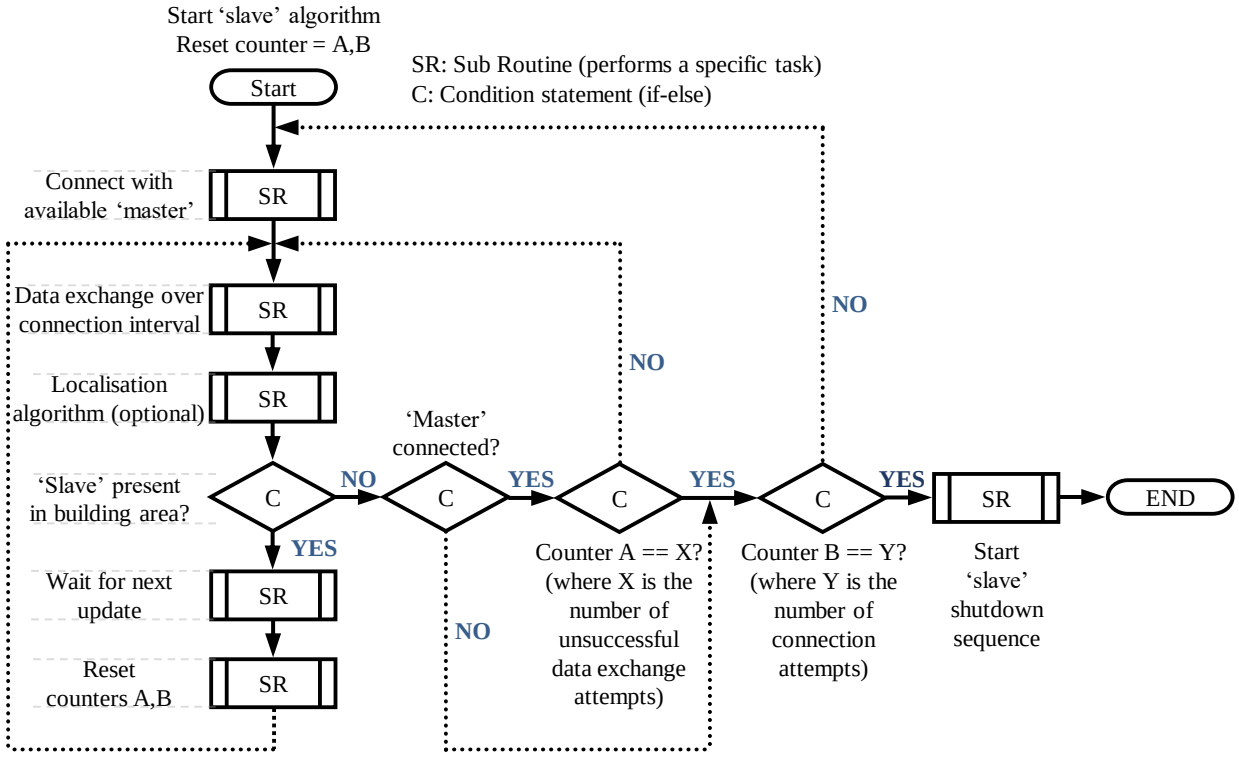


Figure 2.9: Proposed device shutdown algorithm running on device modified from the preliminary study van Rensburg [20, p.6, Figure 2a].

interfaced with the application layer of the BLE protocol stack. The application checks for potential connections by the advertising tablet to any available master nodes located in a building area.

The master node will initiate a connection and the data exchange process commences when a device is in range. Once the maximum of 12 devices are connected to a master, it won't be possible to connect anymore PCDs to it, and the PCDs will have to try to connect to other available masters. However, if a master node reached the maximum number of connected devices and no alternative master node is available and the master has no other nodes in reach for connection, then currently connected devices will be disconnected to allow other devices to connect to a master and register on the network. Note that the slave node will be a portable device such as a tablet while the master node is a fixed node mounted at strategic locations in a large building area. If a device is unable to connect after a predetermined number of unsuccessful attempts, a shutdown procedure of the operating system is initiated by a proprietary application running on the device.

The BLE standard only provides a star topology where multiple devices can connect to standalone master nodes, and no provision is made for interconnected mesh communications between master nodes and ultimately to a $CMS_{\mathbf{2}}$. However, Bluetooth Mesh [36] is a recently developed wireless technology that can be deployed for mesh communications between nodes that work in conjunction with the BLE standard. A $CMS_{\mathbf{2}}$ connected through a cloud service may provide information about device localisation, anomaly detection and connections in the network for remote control and monitoring over the internet. To accomplish this, a master node must also be able to run a second protocol stack with mesh network capabilities using an IP addressable networking technology. In the upcom-

ing sections, an IEEE 802.15.4 [19] build wireless mesh technology is presented with IoT capabilities.

2.4 OpenThread

OpenThread is developed by Google Nest in 2015 and is an open-source implementation of the Thread networking protocol. OpenThread (hereafter referred to as Thread) is a certified Thread component which adheres to all the features defined in the Thread Specification v1.1 [37]. Thread is designed as an IP based low-power wireless mesh networking protocol operating on the IEEE 802.15.4 [19] hardware layer that enables cloud communications over the internet. The ThreadGroup is an alliance of technology companies that are contributing members of the Thread wireless technology that provides Thread certified products. SoC manufacturers such as ARM mbed OS (NXP FRDM-K64F + Atmel ATZB-RF-233), Nordic Semiconductor – nRF52840 (OpenThread), Silicon Labs Mighty Gecko Module MGM12P and TI OpenThread (CC2652) incorporates the Thread protocol stack on their embedded platforms. The University of the Witwatersrand is currently a research member of the ThreadGroup [38], which allows the ICPP group to research Thread and have access to the Thread specification. A company would require an annual membership fee when building solutions and selling products using Thread. The technology is initially designed mainly for automation applications in commercial as well industrial environments through smart systems to advance and accelerate the adoption of the IoT.

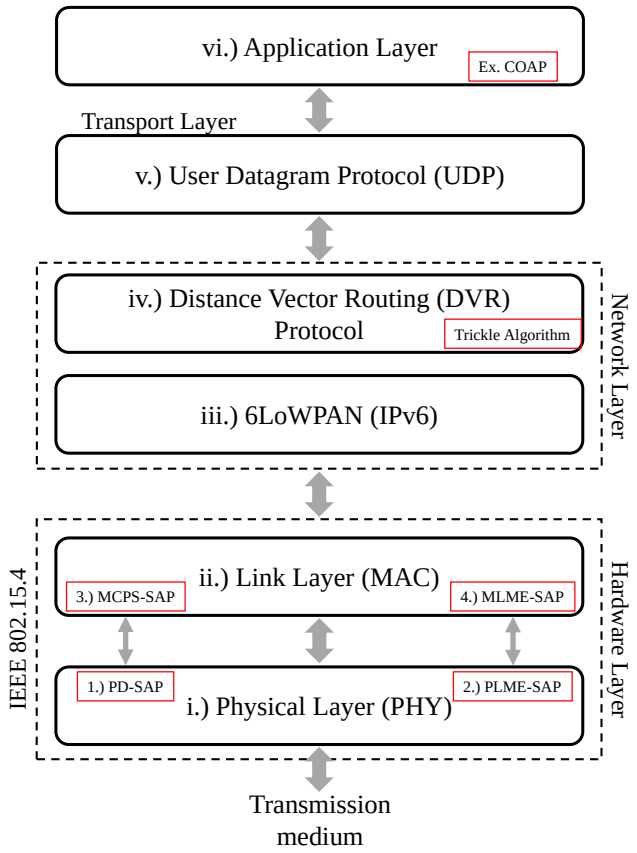


Figure 2.10: Thread protocol stack built on the IEEE 802.15.4 [19] hardware layer.

2.4.1 IEEE 802.15.4

Based on the IEEE 802.15.4 [19] standard, the hardware layer may be designed that either operate using a star or peer-to-peer topology. A peer-to-peer topology allows for nodes within communication range to form complex mesh formations. The mesh topology allows the routing of packets from a transmitter to any receiver through multi-hopping using interconnected nodes. The routing of packets in mesh network topologies is dealt with in the network layer that forms part at the higher levels of the stack and is not covered

in the standard. The hardware layer is broken up in two sub-layers, namely the PHY and MAC layer. In summary, the PHY layer provides the following: Control of the radio transceiver, RF Energy Detection (ED) system, Link Quality Indicator (LQI), channel selection, Clear Channel Assessment (CCA) and modulation and demodulation of the baseband signal. The MAC forms part of the link layer of the OSI model and is mainly responsible for structuring, encapsulation and decapsulation of frames and ensuring reliable packet transmissions through a packet collision avoidance mechanism.

Concerning the lower layers in **Figure 2.10**, the PHY and MAC communicate through what is known as a Service Access Point (SAP). An SAP acts as an interface standard between the layers. The PHY provides the Physical Data - Service Access Point (PD-SAP) and Physical Layer Management Entity - Service Access Point (PLME-SAP). Both these SAPs are used to interact with the MAC layer. The MAC layer provides the MAC Common Part Sublayer - Service Access Point (MCPS-SAP) and the MAC Layer Management Entity - Service Access Point (MLME-SAP) and used to interface between the PHY and the network layer [19, p.245]. Data packets are referred to as the MAC Protocol Data Unit (MPDUs). The IEEE 802.15.4 [19] SAPs consist of the following primitives:

1.) PD-SAP:

Consists of a single primitive named PD-DATA and used for passing MPDUs from the MAC to the PHY [28, p.79].

2.) PLME-SAP:

Allows for management and coordination of a database with the MAC layer that provides the following primitives [28, p 80-81]:

- PLME-CCA:

CCA for Carrier Sense Multiple Access - Collision Avoidance/Detection (CSMA-CA/CD) operations before transmission of an MPDU.

- PLME-ED:

  Energy Detection (ED) measurement to determine the energy level of a channel and then decide which channel to transmit on.

- PLME-GET:

  Used to read information from the PHY PAN Information Base (PIB) that contains metrics such as the channel number and transmit power.

- PLME-SET:

  Used to set PHY PID attributes. For a complete list of PHY PIB attributes, see [19, p.404-410].

- PLME-SET-TRX-STATE:

  Set the state as transmitter or receiver but can only set one state at a time.

3.) MCPS-SAP:

Supports the transport of data to another device known as the MAC data service. See a list of primitives and functions at [19, p.329-338].

4.) MLME-SAP:

Manages the transport of the management command between the network layer and the MLME. See a list of primitives and functions at [19, p.248-327].

The experimental study of this research use (through an API) the MLME-SAP with some of the following MLME-GET, MLME-SET, MLME-COMM-STATUS and MLME-SCAN primitives to conduct link-budget analysis in the deployment of the mesh network. Link-budget analysis deals with determining the signal gains and losses in links between

communicating nodes by measuring received signal strength, estimating signal attenuation through models to calculate RF propagation path lengths and increasing transmitted signal strength of the antenna to increase network performance [39]. The PHY PIB attributes are accessed through the MLME-GET and MLME-SET primitives. Link-budget analysis is performed in some form or other by system engineers in the development of wireless mesh networks [40]. The IEEE 802.15.4 [19] PHY and MAC layers operating on the 2.4 GHz ISM band and network layer are dealt with more comprehensively in the next section.

2.4.2 Protocol stack

With reference to **Figure 2.10**, the layers of the Thread protocol stack consists of the 'hardware', 'network', 'transport' and 'application' layer:

i.) Physical Layer (PHY):

The PHY layer operates at the 2.4 GHz ISM frequency band and uses the Offset-Quadrature Phase Shift Keying (O-QPSK) technique for modulating and demodulating the IF baseband signal. Unlike GFSK, O-QPSK uses two bits per symbol and offer higher data throughput than single bit symbol modulation schemes such as BFSK (GFSK) and BPSK. With reference to the O-QPSK modulation block diagram illustrated in **Figure 2.11**, a 32-bit binary data sequence s_k[12] generated by the MCU is first demultiplex from a serial binary data stream to a parallel even-numbered I_k[13] and odd-numbered Q_k[14] discrete sequence. The data sequences in the

[12] A binary sequence $s_k = \{11, 01, 10, 01, 11, 00, 00, 11, 01, 01, 00, 10, 00, 10, 11, 10\}$.
[13] Results in a mapping of s_k to $I_k = \{1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1\}$.
[14] Results in a mapping of s_k to $Q_k = \{1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0\}$.

continuous-time domain determined by the even and odd-numbered bits respectively are presented as [32, p.168]:

$$I(t) = \sum_{k=-\infty}^{\infty} I_k p[t - kT_s],\ Q(t) = \sum_{k=-\infty}^{\infty} Q_k p[t - kT_s], \tag{2.7}$$

where a logic one and zero from the binary sequence s_k is first converted to the NRZ coding scheme that results in $1 \rightarrow 1$ and $0 \rightarrow -1$, I_k and Q_k is the even and odd binary sequence mapped form s_k respectively and $p(t)$ is a rectangular pulse shaping function of the NRZ converted binary data T_s apart. A half-sine pulse shape technique is employed of the mapped even I_k and odd Q_k sequences before modulation to prevent spreading (limits the bandwidth of the transmitted pulse) of higher and lower frequencies represented by side lobe energies causing interference. The continuous-time[15] representation of the half-sine pulse filter defined by IEEE 802.15.4 [19, p.414] is given by the following function as:

$$p(t) = \begin{cases} \sin\left(\dfrac{\pi t}{2T_s}\right) & , 0 \leq t \leq 2T_s \\ 0 & , \text{otherwise.} \end{cases} \tag{2.8}$$

The I and Q phases BPSK signals are summed to produce the O-QPSK output signal. The O-QPSK incorporates a delay in the Q-channel resulting in the output signal [32, p.180]:

[15] Analog to digital signal conversion (zero-order hold of analog input signal): $x_k = x_c(t)|_{t=kT_s}$.

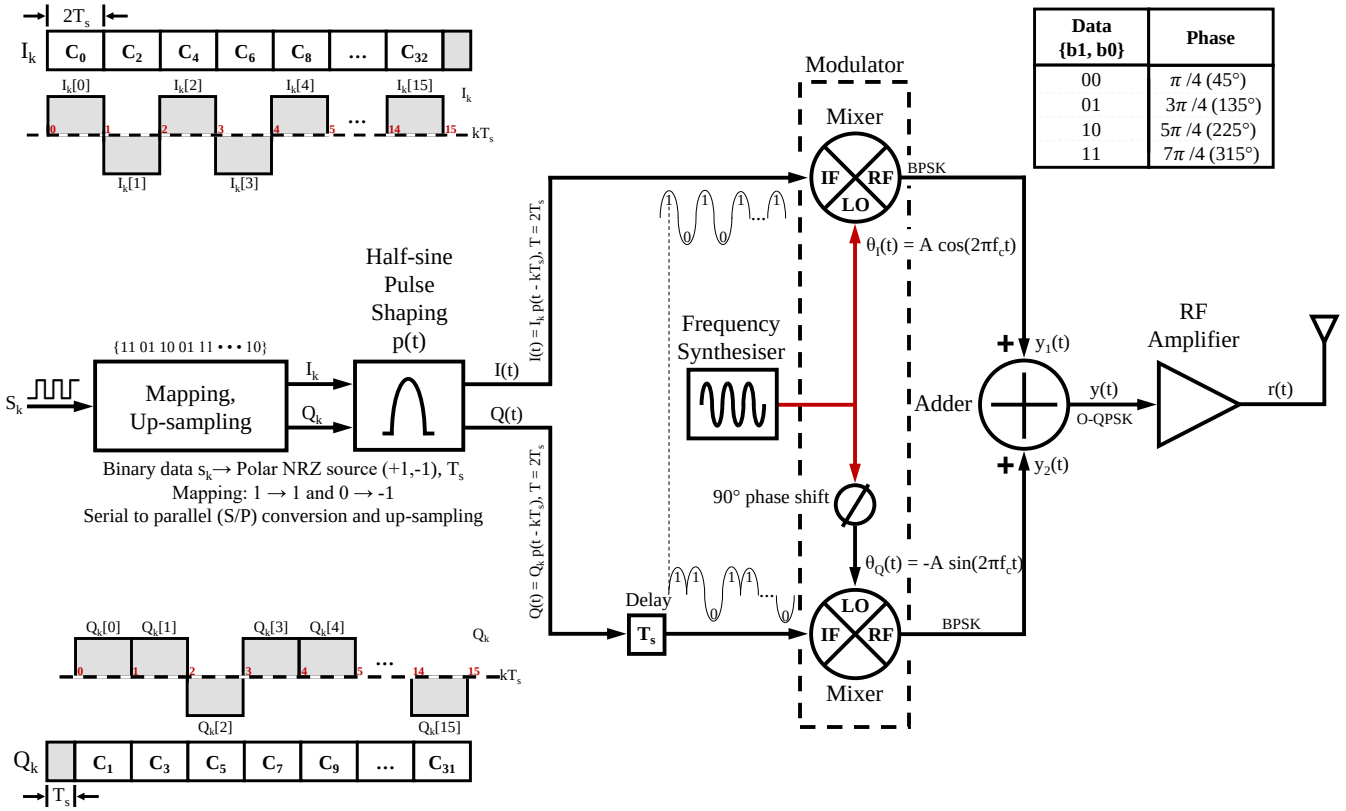


Figure 2.11: Conceptual O-QPSK modulation modified from Xiong [32, p.180].

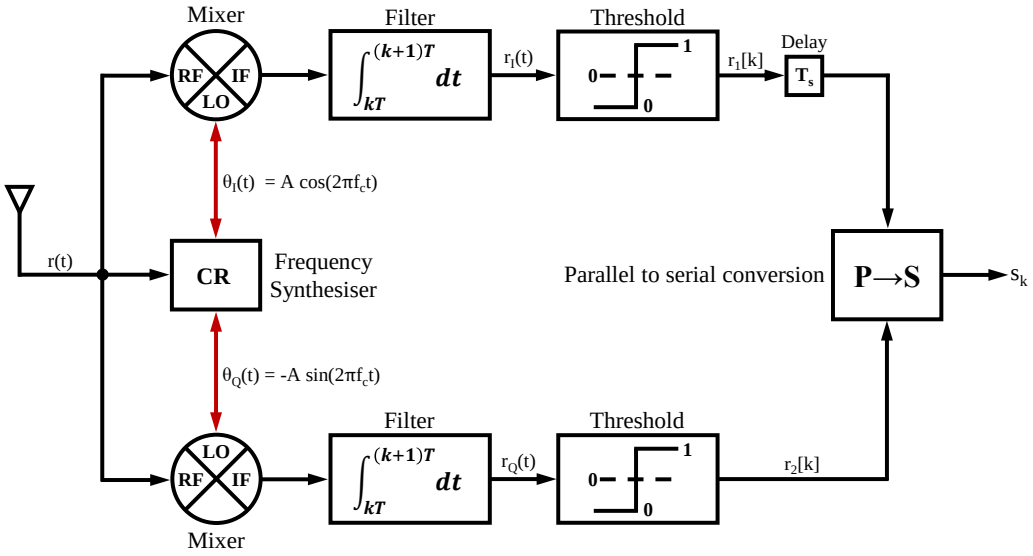


Figure 2.12: Conceptual O-QPSK demodulation modified from Xiong [32, p.181].

$$y(t) = \frac{A}{\sqrt{2}} I(t)\ \cos(2\pi f_c t) - \frac{A}{\sqrt{2}} Q(t - \frac{T}{2})\ \sin(2\pi f_c t), \quad -\infty < t < \infty. \tag{2.9}$$

From **Equation 2.9**, it is evident that O-QPSK transmits data twice as fast as the binary modulation schemes. With reference to **Figure 2.11**, the O-QPSK signal output: {11} with a 315° phase T_s apart, {01} with a 135° phase T_s apart, {10} with a 225° phase T_s apart, {01} with a 135° phase T_s apart, and so forth. **Figure 2.12** demonstrates coherent O-QPSK demodulation. A Carrier Recovery (CR) circuit ensures the transmitted signal is synchronised in frequency and phase with the receiver. The PLL achieves transmitter and receiver synchronisation in the FS. The output signal from the mixer is integrated and pass through a threshold to determine whether a logic one or zero is received. The I and Q phases, presented as the discrete $r_I[k]$ and $r_Q[k]$ sequences, are demodulated separately as BPSK signals and multiplexed to combine the two sequences into a single discrete sequence s_k. The PHY is also responsible for channel selection using the Direct Sequence Spread Spectrum (DSSS) technique. The implementation of the functions such as the mixers (modulation and demodulation), filtering and FS can be done using an Application Specific Integrated Circuit (ASIC) design on physical hardware or using Software Defined Radio (SDRs). SDRs provides more flexibility and may be configured in real-time through software to adapt to various standards and modes of operation. SoC's are available with digital signal processing capabilities where software-defined solutions may be implemented. The minimum hardware configuration required for

SDR's is demonstrated in [41, p.1-2].

ii.) Link Layer (MAC):

Ensure reliable transmissions between nodes over the medium by using CSMA-CA. CSMA-CA use an energy detection mechanism to listen if a channel is clear before modulation occurs. The algorithm is used before the transmission of data and described in detail at IEEE 802.15.4 [19, p.63]. Packets are encapsulated and encrypted at the MAC's Logical Link Control (LLC) sub-layer.

iii.) 6LoWPAN (IPv6):

IPv6 low-power wireless personal area networks protocol that forms part of the network layer and allows IPv6 packets to be sent and received over the IEEE 802.15.4 [19] network. The 6LoWPAN standard was developed for constrained devices that operate on limited memory and processing capabilities.

iv.) Distance Vector Routing (DVR):

DVR forms part of the network layer and is used primarily to relay packets between interconnected nodes. The protocol uses the Bellman-Ford equation implemented in **section 4.1.6** to find the least cost of a link before the routing of a packet. Link cost, in turn, uses the signal strength of neighbouring routers as one of the metrics to determine the shortest distance to a destination router. The routing protocol distance is associated with the number of hops a packet needs to travel from a source to destination. The mesh is continually updated by periodically sending routing costs to neighbouring routers by the Trickle algorithm [42]. The DVR algorithm for implementation on the network layer is illustrated in Forouzan [29, p.598-602].

v.) User Datagram Protocol (UDP):

UDP forms part of the transport layer and is used to send messages between nodes in the network. The protocol is connectionless and used for constrained devices, meaning unlike TCP/IP, no handshaking or acknowledgement of receipt of a packet is performed from the receiver to the transmitter.

vi.) Application Layer:

Thread does not define any protocol for its application layer. However, for low-power constrained devices, the COAP is the most widely used protocol for sending and receiving packets over a network and secure cloud communications. Alternatively, UDP messaging may be directly used from the transport layer for messaging between nodes in a network. Third-party application layer software solutions for IoT connectivity is also available to developers such as Dotdot, KNX and OpenWeave.[16]

2.4.3 Network topology

Leading to the HTEL anti-theft conceptual design, an IEEE 802.15.4 [19] based mesh network topology with IoT capabilities is proposed in **Figure 2.13**. The network topology is also applicable to control and monitoring applications found in the industry where multiple mesh networks are distributed to cover large areas that require instrumentation connectivity through a $CMS_{\mathbf{2}}$. For example, IoT applications at the LHC in monitoring and control of intelligent embedded systems in the field that communicate over a cloud computing service. An intelligent embedded system houses an SoC (acting as a router) and

[16] www.threadgroup.org/What-is-Thread/Developers.

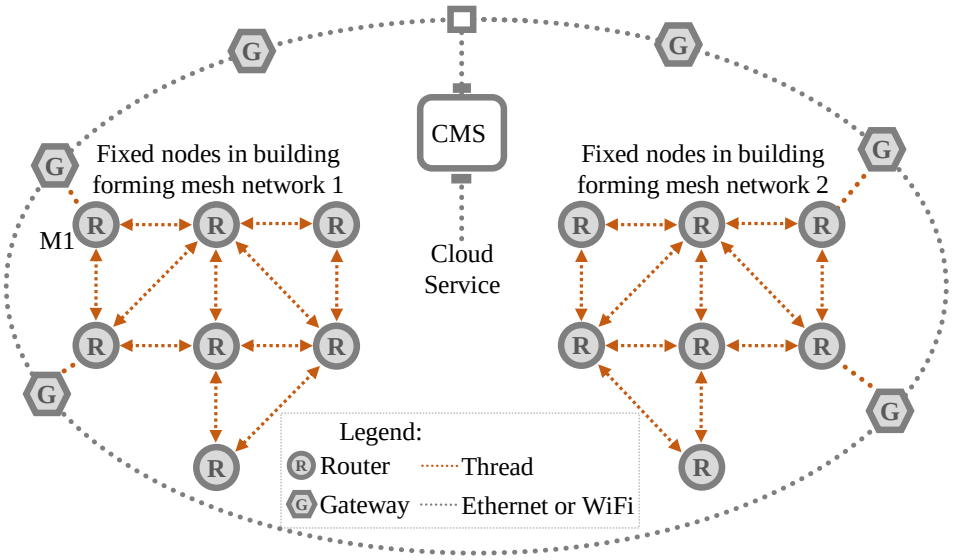


Figure 2.13: Thread network topology proposed for the HTEL anti-theft solution illustrating how multiple IEEE 802.15.4 [19] based mesh networks are connected through redundant gateways over the internet to a cloud service. The Thread end-device nodes are excluded from the network design.

can connect smart sensors while running computationally intensive algorithms as found in machine learning and image processing. The proposed mesh network operates on the IEEE 802.15.4 [19] based PHY and MAC layers that uses the Thread protocol in the networking layer. A Thread network contains nodes that may be configured as border-routers, routers and end-devices.

Border-routers are gateways that connect Thread to an Ethernet or WiFi network for remote monitoring and control over the internet. A border-router is connected to a router to gather information on connected devices or instrumentation through the mesh. Multiple border-routers may exist in a ring topology using an external network that adds redundancy in the gateway system for network communications to the CMS_2 or cloud. If one border-router fails, then another take over the responsibility for network communications. A

cost-effective approach would be to connect at least one border-router per mesh network.

Routers periodically communicate to other routers to update the mesh topology by using routing algorithms (see **section 4.1.6**). End-devices can trigger events of interest from connected sensors and data can be relayed through the mesh to the border-router. One of the routers is always selected as a leader and is responsible for joining other routers in the network.

An end-device is a node that can be used as a sleep device that operates under small MCU processing duty cycles. Sleep devices draw significantly less power than routers resulting in years of operation using CR2032 batteries. End-devices can only communicate directly with their associated router parent to other routers or end-devices in the network. Conversely, nodes acting as routers are not designed as sleep devices according to the Thread Technology and not usually battery powered. A maximum of 512 end-devices are allowed to connect in a star topology to a parent router which could be smaller depending on the amount of RAM available on that router. End-devices are not currently used in the proposed network topology in **Figure 2.13** but are more relevant to low-power sensory applications that require intermittent monitoring or control to a $CMS_{\mathbf{2}}$. ZigBee, which is also an IEEE 802.15.4 [19] based wireless mesh technology contains a PAN coordinator, coordinator and device. A coordinator is similar to a router; the PAN coordinator similar to the leader; a coordinator to a router and a device to an end-device. All routers in the Thread network share persistent data of the leader, if the leader fails, another router node will be selected as a leader that ensures no single point of failure. Thread currently only allows a maximum of 32 routers per mesh network. Therefore, a multiple mesh network may be set up to expand network coverage in a large campus or building area. The Thread

Specification v1.1 [37] outlines a comprehensive technical overview of the Thread wireless technology. To use BLE and Thread on a single SoC, the SoC has to provide multi-protocol capabilities that can run both the stacks concurrently.

2.5 IEEE 802.15.4 Proprietary

Proprietary network developments build on the IEEE 802.15.4 [19] hardware layer is also possible that exclude the use of commercial low-power mesh wireless technologies such as Thread or ZigBee. This may be of specific interest to researchers or designers if a proprietary network layer needs to be developed for an application-specific network solution. However, Thread and ZigBee that natively support IP are considered the preferred low-power mesh wireless technologies by design engineers because of the extensive research and development conducted by companies and the technologies ability to easily connect to the internet.

2.6 Conclusion

In **Chapter 2**, a technical overview was given of the BLE and Thread protocol stacks where special emphasis is placed on the hardware layer. The PHY and MAC were presented while illustrating the techniques responsible for the wireless transmission of electromagnetic signals. These wireless technologies is of special interest to the researcher who is interested in developing novel or improved algorithms on the IEEE 802.15.4 [19] PHY or MAC and

the implementation of the algorithms using either simulation, on ASICs or SDR designs. BLE is the most widely used low-power wireless technology found in smart-devices today, while Thread natively supports IPv6 and provide autonomous mesh formations with IoT capabilities. The HTEL group at the University of the Witwatersrand is an active research member of 'ThreadGroup' and currently researching low-power wireless mesh technologies [43]. Further research may potentially lead to proprietary mesh network solutions running the IEEE 802.15.4 [19] hardware layer in future developments. The engineering effort in designing and development of proprietary networks would be immense, but may lead to the development of innovative and novel applications in IoT. An API is used to configure and access the hardware, network and transport layers in the development of a Thread network. For the HTEL anti-theft solution, it was realised that a multi-protocol SoC is needed that can run both the BLE and Thread stack concurrently. Furthermore, an algorithm needs to be developed that uses the API of the SoC to conduct performance and reliability tests. The tests will form an integral part before application deployment of an anti-theft solution or an instrumentation application that uses the Thread networking protocol. Custom software will be developed to transmit packets between interconnected nodes forming the mesh and log performance and reliability metrics without the use of a cloud service. Consequently, the literature review resulted in the development of the final HTEL anti-theft conceptual network design in **Figure 2.8** for PCD communications to the mesh and **Figure 2.13** for mesh communications to the cloud. The Thread mesh network topology, however, can be used for almost any type of low-power and low-throughput wireless control and monitoring application with the need for IoT connectivity.

Chapter 3

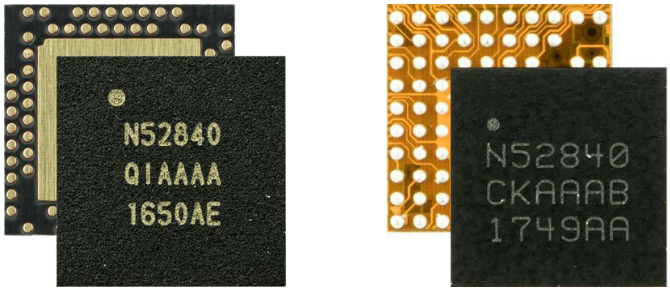
Mesh Network Development

In this chapter, experimental studies are conducted that lead to the development of a mesh network. Following the comprehensive technical overview of low-power wireless technologies presented in **Chapter 2**, an IEEE 802.15.4 [19] based Thread network is developed and implemented. A detailed description of the hardware and software developments are given in the experiments conducted.

3.1 Methodology

An embedded system with a multi-protocol radio with advanced processor capabilities from a silicon chip manufacturer is utilised during the inception phase in network development. An ad-hoc[1] wireless network consisting of nine nodes are strategically distributed in an indoor building environment. Point-to-point packet transmissions from each node in the

[1]Wireless ad-hoc mesh networks are mobile and dynamic communicating interconnecting nodes that do not necessary rely on external network infrastructure to function.



(a) 7 mm × 7 mm aQFN73 package.

(b) 3.5 mm × 3.6 mm WLCSP package.

Figure 3.1: Nordic Semiconductor's a) aQFN73 and b) WLCSP nRF52840-SoC capable of running advanced multi-protocol Bluetooth 5, Thread and ZigBee wireless technologies on a single chip [45].

network are conducted whereby Round-Trip Times (RTTs) and packet loss are recorded. A statistical approach is followed in data analysis to determine the performance and reliability of the mesh network. The results obtained from the statistical analysis are compared with a simulation of the wireless technology routing protocol to confirm the hop distance a packet travelled.

3.2 Experimental Study

The physics building at the University of the Witwatersrand is used as an indoor testing environment where nodes are distributed at specific areas located on the ground floor and mezzanine. The building consists of many obstacles, termed 'shadow regions' caused by obstructions, which may inadvertently affect radio transmissions between the nodes in the network. Environmental conditions for planned network installations at educational institutions in outdoor environments are probably less obstructive than the test-bench en-

vironment, and better network performance are expected. Nodes are mounted on Acrylonitrile Butadiene Styrene (ABS) enclosures at locations determined from the experimental study in **section 3.2.2**. ABS plastic has a relatively small dielectric constant [44] (the ability of the material to build up an electrical charge) and is transparent to RF signals. The plastic may be used as an enclosure to mount sensitive radio transceiver devices for ingress protection against dust and water. The antenna of a node is positioned to avoid absorption of Radio Frequency (RF) energy by the enclosure and to ensure radiation of the omnidirectional RF-modulated signal. The network development is discussed in **section 3.2.3**.

3.2.1 Embedded system and RF design

3.2.1.1 Hardware development

The nRF52840 [45] from Nordic Semiconductor (hereafter referred to as the SoC), is an integrated MCU and multi-protocol 2.4 GHz radio for use in low-power wireless applications. The MCU consists of a 32-bit, 64 MHz, 1 MB flash memory and 256 kB RAM (additional 48 digital/analog IOs, I2C, I2S, SPI, UART and USB communication interfaces) ARM Cortex-M4F processor. The SoC is presented in **Figure 3.1a** and fabricated out of silicon with a case dimension of 7 mm $\times$ 7 mm $\times$ 0.85 mm containing 55 nm transistors, resulting in a processor performance of a 212 EEMBC CoreMarks score [46] or 3.31 Coremarks/MHz running from flash memory. The PCB design embeds an advanced Quad Flat No-lead (aQFN) package (SoC), passive components and trace antenna to represent a network node.

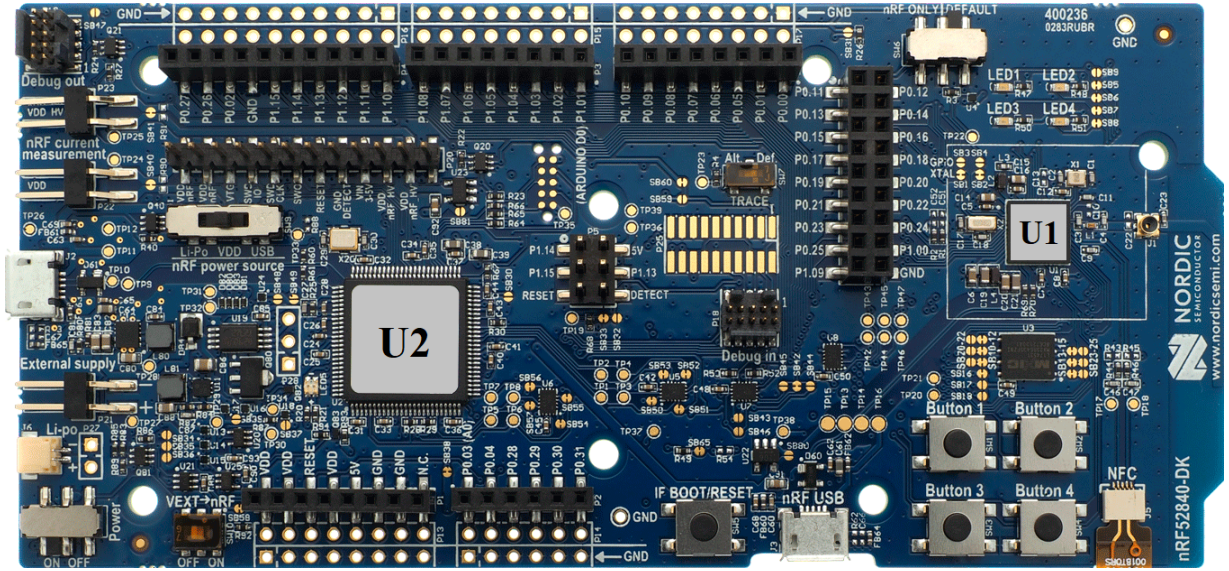


Figure 3.2: nRF52840-DK [48] utilised in the experimental study for fast prototyping and acting as a router in the Thread network. The nRF52840-SoC (labelled U1) with the passive components is mounted on the top layer located inside the square silkscreen marking. The rest of the nRF52840-DK PCB embeds an MCU (labelled U2) for programming and accessing IO ports of the nRF52840-SoC.

An aQFN is a leadless pad, multi-row, fine pitch lead frame packaging solution. The aQFN uses more material, offer PCB design advantages and is more expensive than Water-Level Chip-Scale Packages (WLCSP). WLCSP is a leadless solder balls, multi-row, fine pitch with no frame package solution. In the nRF52840 Specification v1.1 [47] release, a smaller WLCSP package, depicted in **Figure 3.1b**, is designed with the same chip architecture as the aQFN73 package.

The nRf52840-DK [48] embedding the SoC and MCU programmer is shown in **Figure 3.2** and utilised in this study for fast prototyping in the development of the mesh network. The SoC is located to the right of the PCB and labelled U1. The second MCU, labelled U2, is used to program the wireless protocol stacks and application to the SoC. The nRf52840-DK provides access to the IOs of the SoC and is accessible through the header pins. A 3.6 V, 1.5 Ah rechargeable Nickel–Metal Hydride (NiMd) battery is used to power the SoC through the external supply connector on the left of the figure running Thread and the testing software.

In the RF design, a matching network called a pi-filter consisting of two capacitors labelled C3 and C4 and inductor L1 to ensure a 50 Ω impedance between the SoC X2 and antenna. The trace of the single-ended monopole antenna located on the top copper layer is designed with a ground plane on the bottom layer shown in of the PCB to form a dipole antenna. The length of the antenna trace is at least one-quarter of the 2.4 GHz wavelength. This acts as a transmission line and will show characteristic impedance due to inductance and capacitance. The 50 Ω impedance, which is a compromise between 30 Ω for minimum attenuation and 75 Ω for maximum power transfer, is a well-known standard used in RF antenna design. The impedance of the antenna trace is tested by

connecting an RF measuring probe J1 to a network analyser, taking into account the effect of the dB loss of the probe into consideration [49]. The goal of antenna tuning is to maximise radiation by resonating the antenna at the operating frequency of the radio. The SoC antenna transmits a maximum output power of $T_x = +8$ dBm or 6.25 mW with an antenna gain approximating 0 dBi that radiates the modulated omnidirectional signal to nearby nodes. The Received Signal Strength Indicator (RSSI) in decibel milliwatts, measured at approximately RSSI ≥ -35 dBm at a node within a 1 m proximity over the air with no obstacles present. To ensure similar signal performance of the nRF52840-DK, it is imperative the antenna guidelines are adhered to when designing the PCB embedding the nRF52840-SoC. Improper impedance matching can cause transmitter power loss, which in turn reduces radio transmitting range [50]. Consequently, the antenna may be susceptible to interference, which will have a detrimental effect on the performance of the RF network. For more information on the PCB and antenna design, refer to **Appendix II**.

3.2.1.2 Software development

The software development process uses an API and a binary representation of the Thread protocol stack that is packaged into the '*nRF52 SDK for OpenThread*' [51] Software Development Kit (SDK) developed by Nordic Semiconductor. The API consists of programming functions to interface with the Thread protocol stack. Custom testing software is compiled with the GNU Arm Embedded Toolchain [52] from GCC (GNU Compiler Collection) [53]. GNU Arm Embedded Toolchain is a GCC compiler used in the software development of ARM Cortex-M4 processors. The testing software sets up the node roles, assigns IPv6 ad-



(a) R_1 and R_2 router. **(b)** R_3 router. **(c)** R_4 router.



(d) R_5 and R_6 router. **(e)** R_7 router. **(f)** R_8 and R_9 router.

Figure 3.3: Physics building ground floor router distribution determined from log-distance modelling. Each router is mounted on an enclosure to ensure radiation of the omnidirectional RF-modulated signal.

dresses and sends Internet Control Message Protocol (ICMP) requests between the nodes for the network performance and reliability tests. The Thread stack and testing binaries are transferred to the nRF52840-SoC using the SEGGER-link programmer tools [54]. The Thread Specification v1.1 [37] presents a comprehensive technical overview of the technology with important aspects discussed in **section 2.4**. As part of this study, the testing software included an algorithm that uses the ICMP to log Quality of Service (QoS) metrics and is discussed in **section 3.2.3**.

3.2.1.3 Power consumption

The power consumption of a router during the network tests running the Thread protocol stack outputting a transmitter power of $T_x = +8$ dBm while performing ICMP requests and external logging on a storage device, did not exceed an average of 20 mA of current on the nRF52840-DK. The life expectancy of the 1.5 Ah rated NiMd battery supplying the nRF52840-DK acting as a router, assuming a battery efficiency of 100% and an SoC core voltage of 3 V, is estimated at $(1500\ \text{mAh} \times 3\ \text{V})(20\ \text{mA} \times 3\ \text{V})^{-1} = 75$ hours, which allows approximately three days of testing of each router in the network at the Physics building. According to the Thread Technology, routers are not initially designed as sleep devices and battery powered. It is estimated that running only the Thread protocol stack acting as a router on the nRF52840-SoC at $T_x = +8$ dBm using the bare minimum of components required in the PCB design (refer to **Appendix II** for more details), may consume more or less an average current of approximately 10 mA. If the power consumption is found unacceptable, then the Thread software can be modified and/or the radio reprogrammed

to ensure lower power consumption. In anti-theft applications, it is recommended that the mesh is continually active in case an event is triggered from low-powered sensors for remote communications to the CMS. Pocket-sized 3.6 V Lithium-Thionyl Chloride (Li-SOCl2) batteries from Tadiran rated at 35 Ah [55] will last approximately 9000 hours at a load of 3.5 mA. End-devices on the other hand, are sleep devices operating at lower duty cycles which results in a significantly less power (< 0.5 mA) usage than routers and can operate years using pocket-sized batteries.

3.2.2 Router distribution

Path losses are signal attenuations of electromagnetic wave propagations due to refection, diffraction and scattering in indoor environments which may be modelled to determine the RF range [56, p.69]. In the Physics building ground floor hall area shown in **Figure 3.3a**, signal strength measurements between two nRF52840-DK transceivers were taken at various distances to determine the RF coverage area before the deployment of the routers. By utilising two transceivers running the IEEE 802.15.4 [19] hardware layer, the RF path loss is determined from the log-distance path loss model [56, p.102]:

$$\text{RSSI} = 10n \log_{10}(d) + e, \tag{3.1}$$

where $RSSI$[2] is the received signal strength indicator in dBm from a transmitting node, n is the path loss exponent, e is an environmental constant and d is the distance of the

[2]RSSI and power in watts conversion: $\text{dBm} = 10\log(\text{mW}/1\text{mW})$ and $\text{mW} = 10^{(\text{dBm}/10)}$.

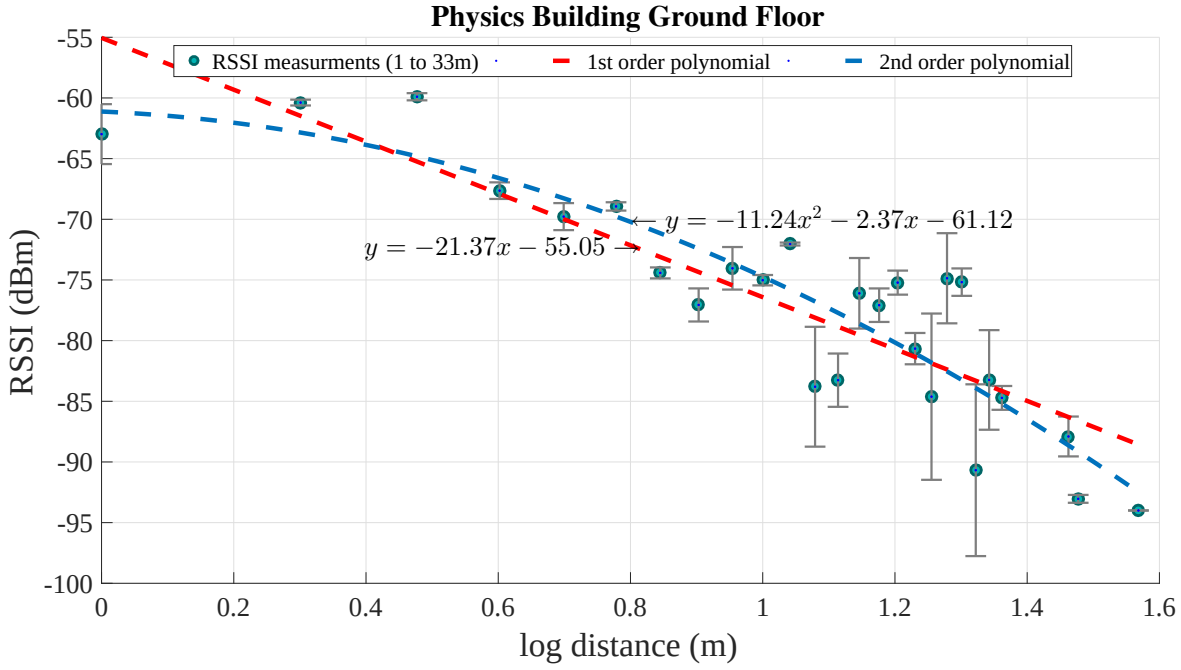


Figure 3.4: Log-distance modelling of RSSI versus $\log_{10}$(d) at $T_x = 0$ dBm.

RSSI measurements between the two nRF52840-DK transceivers in meters. Parameter n is determined using polynomial least-squares regression of the logarithmic distances and averaged *RSSI* measurements. With reference to **Equation 3.1** and the model representation in **Figure 3.4** as $y = mx + c$, the path loss exponent n is estimated at 2.137 where $m = 10n$ and the environmental constant e estimated at 55.05. In free space, $n = 2$ and where obstacles are present, $n > 2$ [56, p.104]. The fit demonstrated 87.58% of the total variation in the data is about the average. By running similar experiments, nodes may be optimally distributed to cover a wide range of areas in the building. RF attenuation due to obstacles may be determined experimentally through similar field measurements. To aid in indoor propagation modelling, researchers have accumulated huge amounts of signal loss measurements through building material in indoor environments [56, p.127].

Further analysis in **Figure 3.4** predicts reliable data transmissions should be expected between transceivers not exceeding $\log_{10}(20)$ line-of-sight distances or $RSSI \geq -80$ dBm in both directions for a node transmitting at $T_x = 0$ dBm. The indoor RF propagation modelling between two routers with a -90 dBm receiver antenna sensitivity and radiating at $T_x = 0$ dBm (1 mW) output power will have an RF signal attenuation of approximately 30 dB over 30 m line of sight distances [57]. These transmitter and receiver measurements revealed a 2.1 path loss exponent obtained from the log-distance path loss model. Consequently, routers may be distributed not exceeding distances 20 meters apart because of strong signal strength fluctuations illustrated in **Figure 3.4** at this range caused by multipathing of the transmitted signal in the hallway. Statistical fluctuations are presently known as jitter round these distances that may cause performance and reliability degradation in data communications between interconnected nodes. In the experimental study, a node is programmatically set at $T_x = +8$ dBm to allow higher RF line-of-sight distances and take advantage of RF reflections and scattering in a highly obstructed or where no line-of-sight communications are possible in the indoor testing environment.

3.2.3 Mesh network development

In network development, nine routers are distributed in the Physics building. Signal strength measurements are first acquired using the API to determine the quality of the router's link connections. The network test-bench with link-costs is illustrated in **Figure 3.5** with router locations displayed in **Figure 3.3**. The network test-bench display distance in meters between routers and averaged *RSSI* in dBm of interconnected routers.

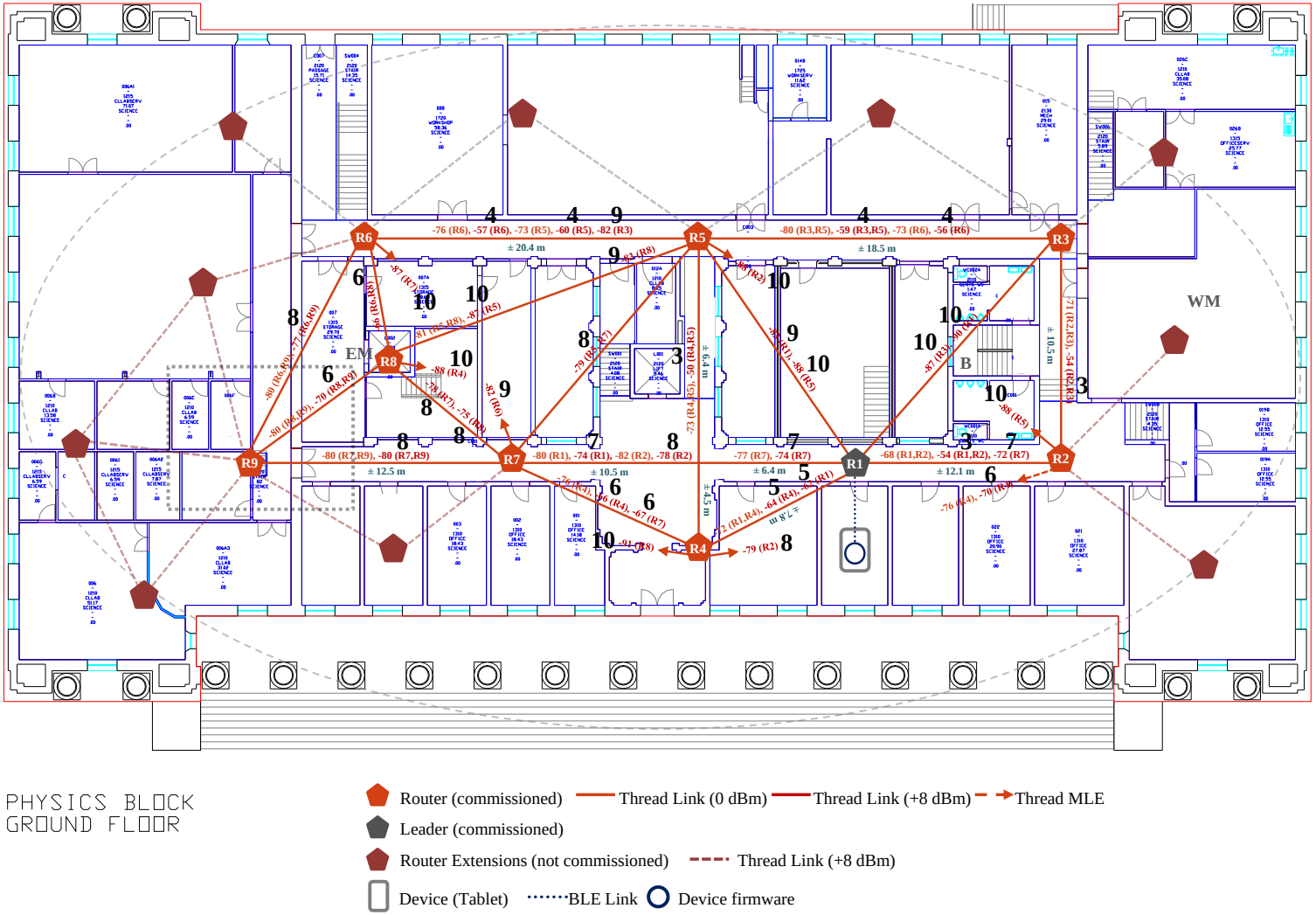


Figure 3.5: Mesh network test-bench development at Physics building.

Table 3.1: Links displaying the average signal strength with link cost between interconnected routers forming the mesh network.

	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**
R1	-33	-53	-87	-64	-88	NL	-74	NL	NL
R2	-54	-35	-54	-70	-88	NL	-72	NL	NL
R3	-90	-58	-30	NL	-59	-56	NL	NL	NL
R4	-62	-79	NL	-31	-51	-90	-67	-91	NL
R5	-83	-88	-61	-50	-23	-57	NL	-83	NL
R6	NL	NL	-82	NL	-60	-26	NL	-66	-78
R7	-74	-75	-78	-66	NL	-82	-29	-78	-74
R8	NL	NL	NL	-88	-87	-63	-75	-28	-73
R9	NL	NL	NL	NL	NL	-77	-80	-70	-26

RSSI:	**Link Cost:**	
No Link	$\leq$ -85dBm	9
$\leq$ -99 dBm	$\leq$ -80 dBm	8
$\leq$ -80 dBm	$\leq$ -75 dBm	7
$\leq$ -70 dBm	$\leq$ -70 dBm	6
$\leq$ -60 dBm	$\leq$ -65 dBm	5
$\leq$ -50 dBm	$\leq$ -60 dBm	4
$\leq$ -30 dBm	$\leq$ -55 dBm	3
	$\leq$ -90 dBm	10

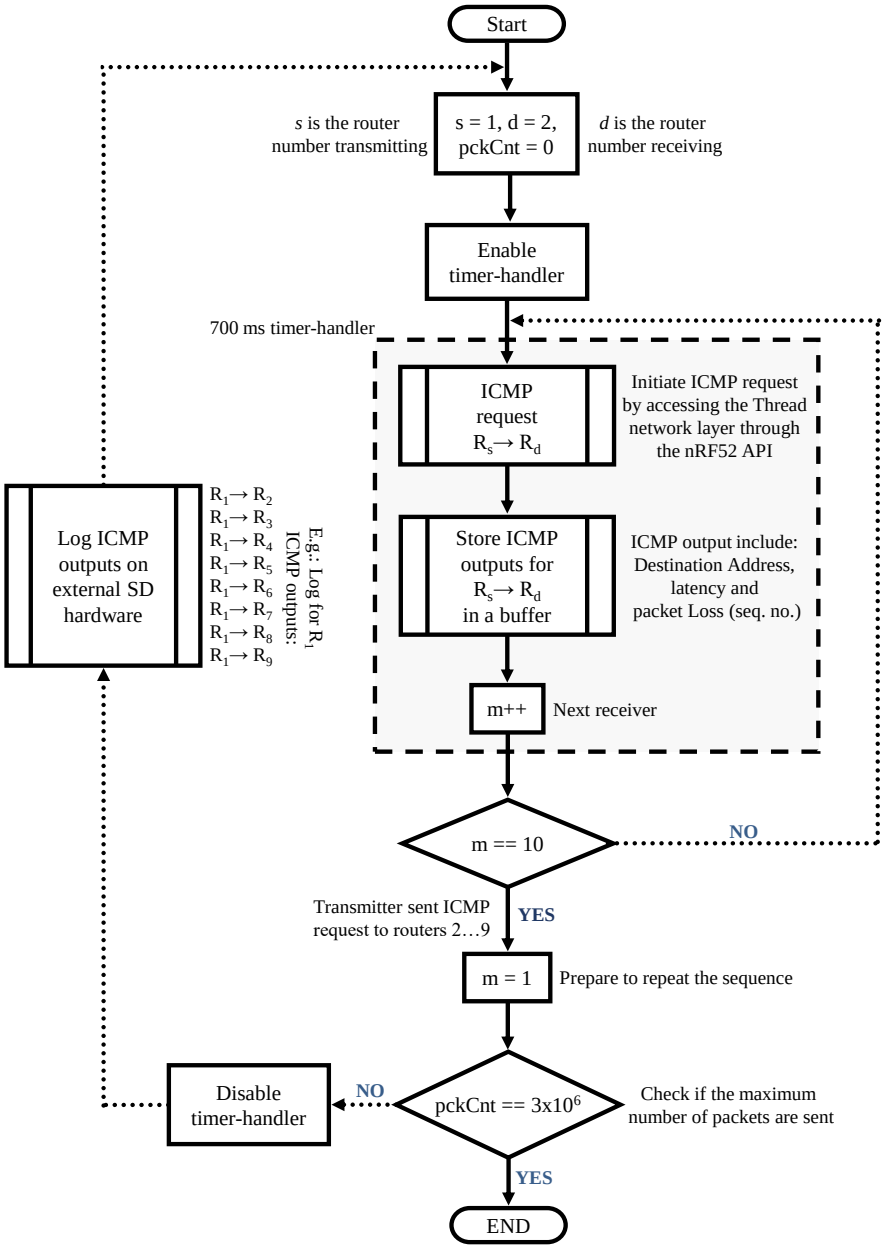


Figure 3.6: Router $R_s \rightarrow R_d$ packet request and logging algorithm.

Mesh Link Establishment (MLE) is a protocol used by Thread where routers periodically multicast an estimate of the quality of their links [37, p.58]. Link-cost is a metric used in this study to determine the quality of a link between routers. *RSSI* measurements between the links of the routers with their link costs are displayed in **Table 3.1**. The Thread specification uses four-link costs ranging from one to four, with one being the best.

The RTT is the time a packet takes to reach the destination router and a return reply of the packet to the source router. RTT and packet loss are obtained by implementing the algorithm in **Figure 3.6** which makes use of the functions in the API to send ICMP requests from a source router R_s where s = 1...9 to the destination routers R_d where d = 1...9. Each router is given a static IPv6 address for data communications and radios are programmed at T_x = +8 dBm. Each router is then randomly activated from a remote nRF52840-DK running Thread to start the algorithm. The timer-handler in the algorithm coded on lines 717-788 (see **Appendix III**) is called by the real-time clock timer of the MCU every 700 ms, which sequentially sent unicast ICMP requests from the source to the destination routers. Logging of an ICMP request is only performed when the timer-handler is disabled and occur just after all the data from a sequence is concatenated on the router's external storage device. Disabling the timer-handler prevents ICMP requests and enables logging of the data by the storage data hardware, which requires the serial peripheral interface of the MCU. Logging is again enabled in a "main()" function and the timer-handler started containing the ICMP request code.

The data logged for analysis in the network, is 42000 × 8 = approximately 336000 ICMP packet requests per router. Since there were nine routers, 336000 × 9, gave a total of approximately 3024000 ICMP packet requests in the mesh network ICMP requests were

transmitted every 0.7 seconds per router, that resulted in 336000 $\times$ 0.7 seconds $\div$ 3600 seconds = approximately 65.3 hours or 2.7 days of logging. The data collected in the study is available for analysis at ICMP Router Data [58].

3.3 Conclusion

A network was successfully developed containing nine routers in the Physics building using Nordic nRF52 SoCs and OpenThread. Link-budget analysis was partially conducted which resulted in the derivation of a log-distance model that aided in the distribution of the routers in the building. An algorithm was developed and flashed to the nRF52840-SoC and randomly activated remotely where ICMP requests were sent from each router acting as a transmitter to all the routers in the network acting as receivers. The network test-bench was presented where the routers are located and mesh links were identified. A detailed view of the hardware and software developments conducted in the study to build the network was discussed. The network test-bench consists of nine source and eight destination routers, therefore 72 routes for data communications. More than three million ICMP request was transmitted, emulating a busy mesh network between the routes and concurrently logging RTT and packet loss measurements over 65 hours on nine external storage devices connected to the nRF52840-DK. A route is a path a packet travels from the source to destination router given by $R_s \rightarrow R_d$ and have two column vectors with n samples containing the RTT and packet loss measurements. The logged performance and reliability metrics of the mesh is analysed in **Chapter 4** to characterise the network.

Chapter 4

Results

In this chapter, the data collected on the external storage devices for each of the nine routers distributed in Physics building shown in **Figure 3.5** are analysed. The statistical techniques used to analyse the data is presented, followed by a detailed analysis of the performance and reliability metrics for each of the routes presented as $R_s \rightarrow R_d$. As mentioned in the previous chapter, network performance analysis deals with round-trip times and hop distances of a route while network reliability analysis deals with the outliers and packet delivery ratio of a route. Conclusions are presented at the end of the chapter giving a comprehensive quantitative overview of the analysed data.

4.1 Data Collection

In the preliminary data analysis presented in van Rensburg et al. [57], unicast Round-Trip Time (RTT) measurements from a single source to multiple destination routers through

multi-hopping were conducted. A total of 9000 packets (1000 ICMP requests per destination router) were sent between a single source R_1 to each destination router R_2 through to R_9, and the RTTs of the payload and packet loss were logged.

In the network development by van Rensburg et al. [26], more than three million packets were sent between multiple source routers to multiple destination routers where performance and reliability data were logged for analysis by adopting similar statistical methods used in particle physics. The ICMP Router Data [58] is collected from the implementation of the algorithm in **Figure 3.6**.

4.1.1 Data analysis

In this section, we analyse the RTTs of a route $R_s \rightarrow R_d$ to compute the simple *three-sigma* limit rule to separate data points to be considered *outliers*. The z-score is used to determine the number of standard deviations an RTT measurement of a route (represents as a vector containing the RTTs measurements) is from the mean. The z-score is given as:

$$\mathbf{z} = \frac{\mathbf{x} - \mu}{\sigma}, \tag{4.1}$$

where $\mathbf{x}$ is a vector containing the RTTs of the route, μ is the mean, σ is the standard deviation of the route $R_s \rightarrow R_d$ and $\mathbf{z}$ is a vector containing the z-scores of the RTT measurements that fall within and outside 3σ. As we recall, a route is presented by $R_s \rightarrow R_d$, where s is the source router and d is the destination router. There's nine

routers in the building, therefore, $9 \times 8 = 72$ possible routes for a packet to travel from a source (R_s) to destination (R_d) router (eight routes per transmitting router where there are nine transmitting routers). The statistical analysis revealed that RTTs do not follow a single Gaussian normal distribution and as a result, a non-parametric statistical model is derived for probability testing. A total number of 72 vectors containing the sampled data are populated containing the RTT measurements within and outside 3σ where their distributions are analysed at a 8-byte packet payload.

4.1.2 Performance and reliability metrics

Outliers can potentially indicate reliability degradation in a route, for example, Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) operations on the IEEE 802.15.4 [19] data link layer that consequently causes packet retransmissions. RTTs within 3σ may provide information on point-to-point hopping distances, while RTTs outside 3σ and packet loss may indicate a route's reliability. A statistical model representing the Probability Density Function (PDF) of the RTTs within 3σ for each distribution is illustrated where the metrics for performance and reliability testing are outlined:

i.) For **performance** analysis, the RTTs of a route $R_s \rightarrow R_d$ within 3σ is separated followed by the computation of the following metrics:

- The mean or average, denoted by symbol μ in ms given as:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{4.2}$$

where n is the number of samples or sample size of the RTTs and x_i is the i^{th} sample of the row vector representing the RTTs within 3σ.

- The median denoted by symbol δ in ms, where the result for each median computation of each route is shown in the upcoming figures.
- The standard deviation denoted by symbol σ in ms and given as:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}|x_i - \mu|^2}, \tag{4.3}$$

where n is the number of samples or sample size of the RTTs and x_i is the i^{th} sample of the row vector representing the RTTs within 3σ. Jitter or variance is computed as σ^2.

- The RTTs of a route is fitted with a non-parametric Gaussian Kernel[1] function given as:

$$f(x,p) = \frac{1}{nh}\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi h^2}}e^{-\frac{(\mathbf{x}-\mathbf{p})^2}{2h^2}}, \tag{4.4}$$

as it was found that a parametric distribution such as a single Gaussian normal distribution cannot properly describe the RTTs. Parameter h is the smoothing parameter or bandwidth = 0.8; n is the number of samples or sample size of the univariate RTTs; $\mathbf{x}$ is a row vector of the RTTs within 3σ and $\mathbf{p}$ is a row vector

[1] A Gaussian Kernel can fit multi-modal distributions for probability testing.

of the bin data. **Equation 4.4** is executed in an iterative loop indexing the i^{th} sample of the route to obtain the PDF (see **Appendix IV**).

ii.) For **reliability** analysis, the following metrics are used:

- The RTTs of a route $R_s \rightarrow R_d$ outside 3σ is considered *outliers* and presented as statistical fluctuations, irrespective whether the RTT distribution follows a normal distribution.
- The RTTs within and outside 3σ, the PDR is obtained as a metric computed from the logged ICMP request of a route $R_s \rightarrow R_d$ and defined as:

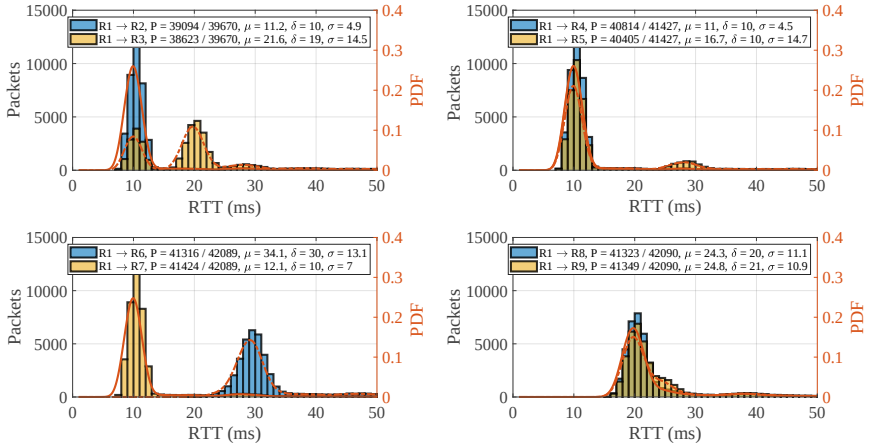
$$PDR_{(R_s \rightarrow R_d)} = \frac{P_r}{P_t} \times 100\%, \tag{4.5}$$

where the PDR is the percentage of packets successfully send from the transmitting router of a route $R_s \rightarrow R_d$, P_r is the total packet received, and P_t is the total packets transmitted.

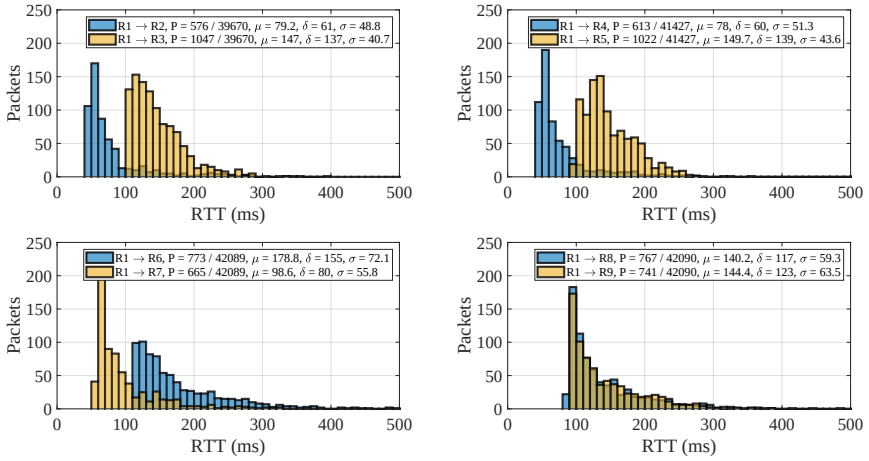
4.1.3 Performance analysis

In this section, we study the RTT distributions of a route $R_s \rightarrow R_d$ using statistical analysis as described in **section 4.1.2**. We first focus on the packet transmissions of router R_1 to its destinations routers by analysing **Figure 4.1** and summarising the results as follows:

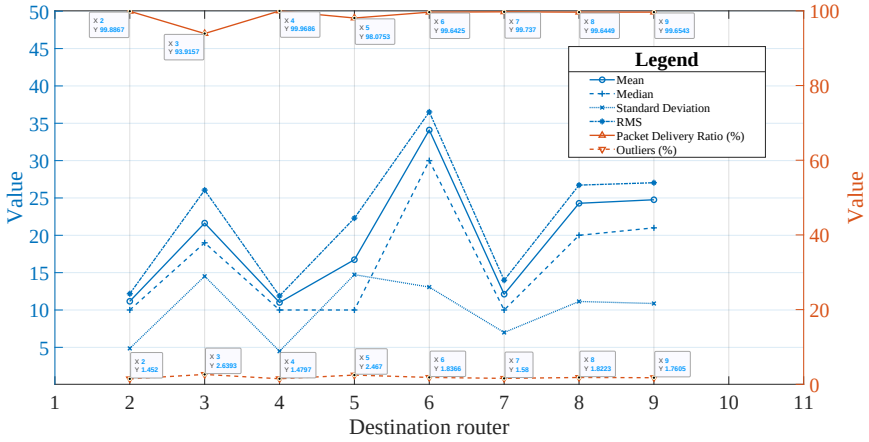
1. Single-link packet transmissions detected in routes $R_1 \rightarrow \{R_2, R_4, R_7\}$:
   i.) No visible performance or reliability issues present in the routes. The bulk of



(a) Router R_1 RTTs distributions within 3σ.

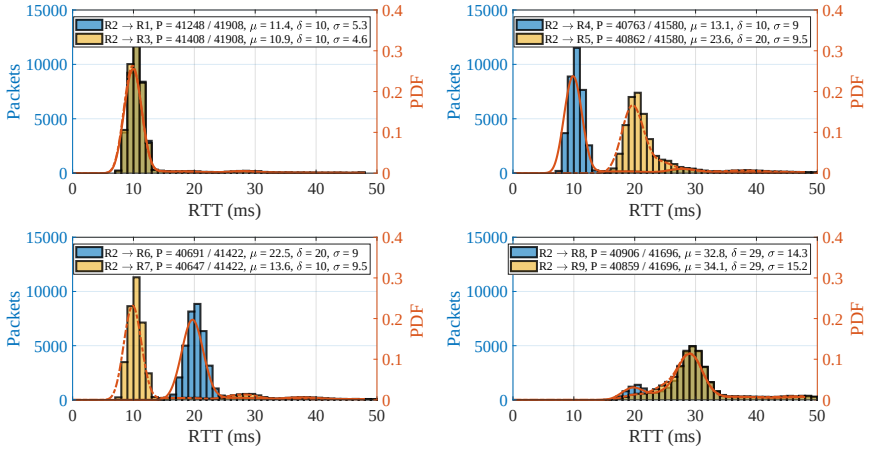


(b) Router R_1 RTT distributions outside 3σ.

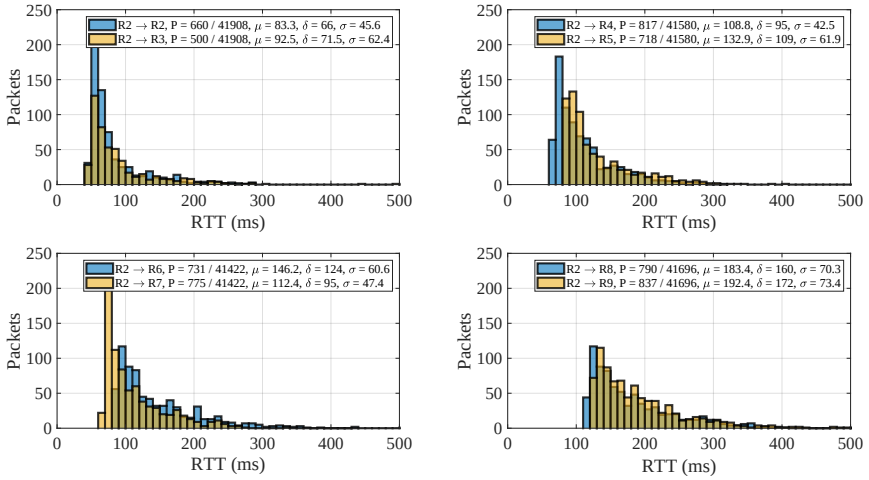


(c) R_1 RTT statistics and packet loss.

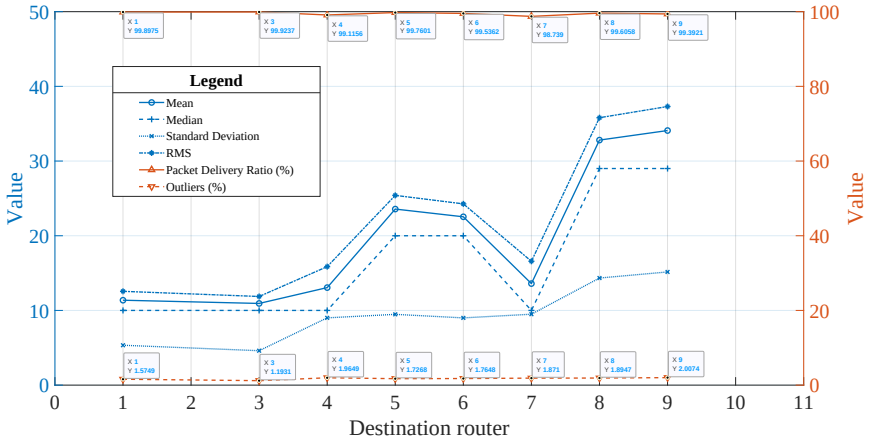
Figure 4.1: Router $R_1 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c) $PDR_{(R1 \rightarrow R2\,:\,42240 \rightarrow 42195)} = 99.89\%$; $PDR_{(R1 \rightarrow R3\,:\,42240 \rightarrow 39670)} = 93.92\%$; $PDR_{(R1 \rightarrow R4\,:\,42240 \rightarrow 42227)} = 99.97\%$; $PDR_{(R1 \rightarrow R5\,:\,42240 \rightarrow 41427)} = 98.08\%$; $PDR_{(R1 \rightarrow R6\,:\,42240 \rightarrow 42089)} = 99.64\%$; $PDR_{(R1 \rightarrow R7\,:\,42240 \rightarrow 42129)} = 99.74\%$; $PDR_{(R1 \rightarrow R8\,:\,42240 \rightarrow 42090)} = 99.64\%$; $PDR_{(R1 \rightarrow R9\,:\,42240 \rightarrow 42094)} = 99.65\%$.



(a) Router R_2 RTT distributions within 3σ.

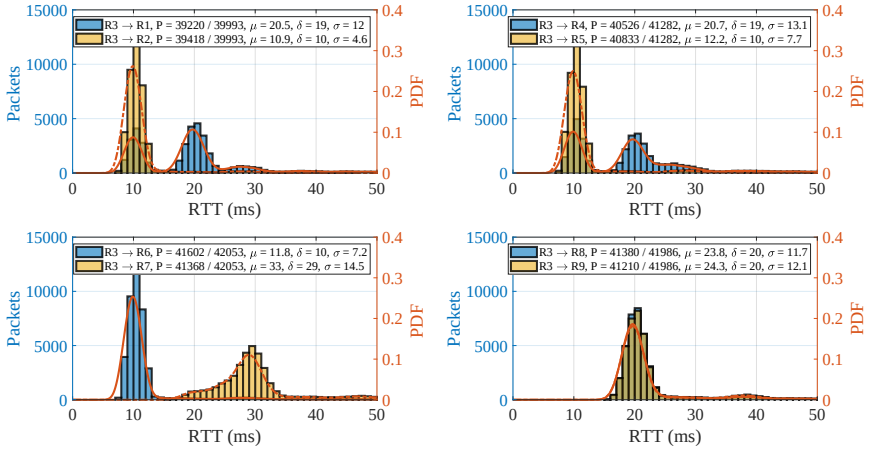


(b) Router R_2 RTT distributions outside 3σ.

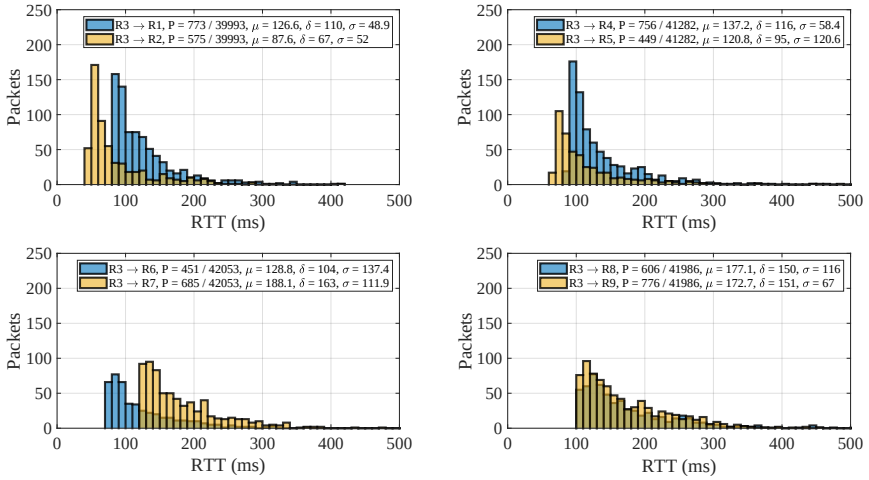


(c) Router R_2 RTT statistics and packet loss.

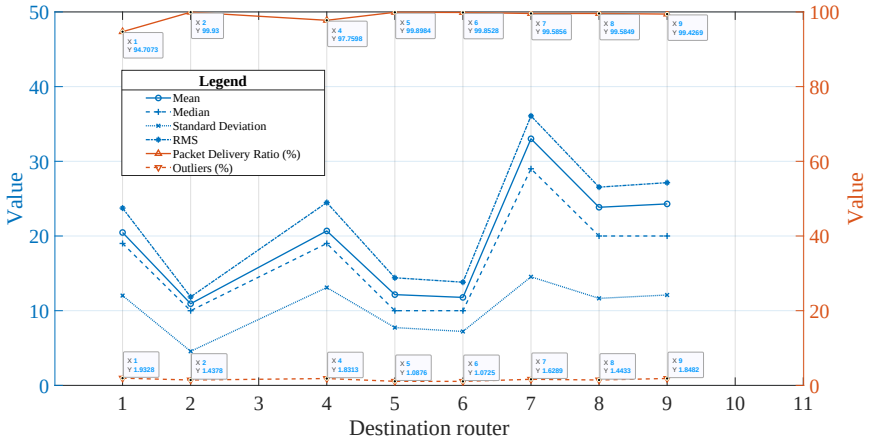
Figure 4.2: Router $R_2 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c)
$PDR_{(R2\rightarrow R1\,:\,41951\rightarrow 41908)} = 99.90\%$; $PDR_{(R2\rightarrow R3\,:\,41951\rightarrow 41919)} = 99.92\%$;
$PDR_{(R2\rightarrow R4\,:\,41951\rightarrow 41580)} = 99.12\%$; $PDR_{(R2\rightarrow R5\,:\,41951\rightarrow 41851)} = 99.76\%$;
$PDR_{(R2\rightarrow R6\,:\,41951\rightarrow 41758)} = 99.54\%$; $PDR_{(R2\rightarrow R7\,:\,41951\rightarrow 41422)} = 98.74\%$;
$PDR_{(R2\rightarrow R8\,:\,41951\rightarrow 41786)} = 99.61\%$; $PDR_{(R2\rightarrow R9\,:\,41951\rightarrow 41696)} = 99.39\%$.



(a) Router R_3 RTT distributions within 3σ.

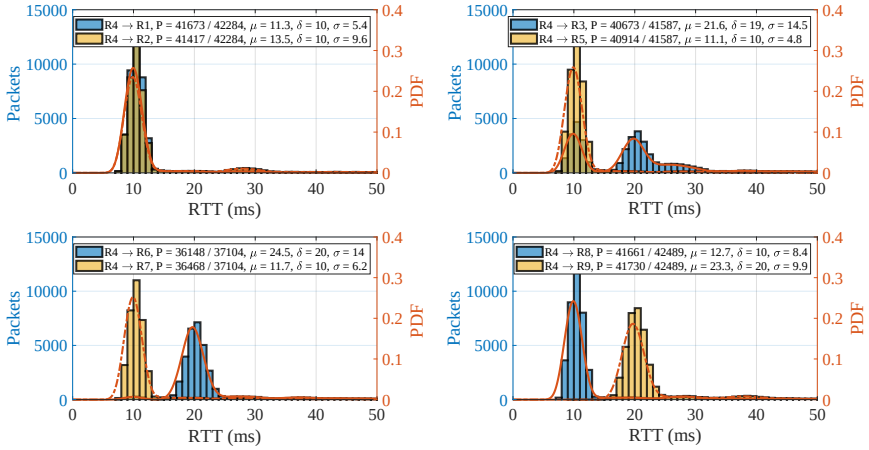


(b) Router R_3 RTT distributions outside 3σ.

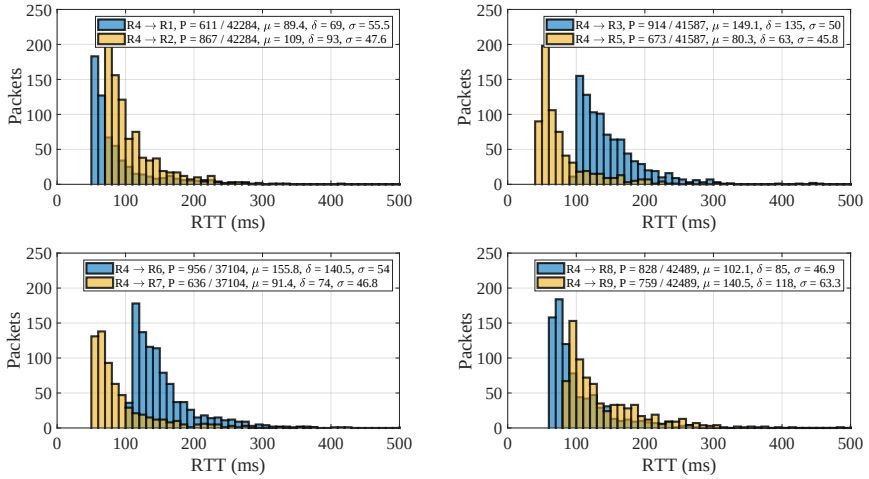


(c) Router R_3 RTT statistics and packet loss.

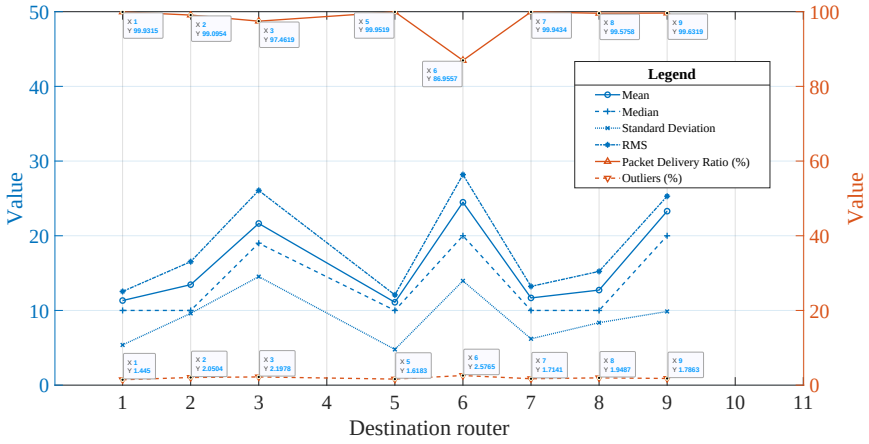
Figure 4.3: Router $R_3 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c) $\text{PDR}_{(\text{R3}\rightarrow\text{R1}\,:\,42228\rightarrow39993)} = 94.71\%$; $\text{PDR}_{(\text{R3}\rightarrow\text{R2}\,:\,42228\rightarrow42200)} = 99.93\%$; $\text{PDR}_{(\text{R3}\rightarrow\text{R4}\,:\,42228\rightarrow41282)} = 97.76\%$; $\text{PDR}_{(\text{R3}\rightarrow\text{R5}\,:\,42228\rightarrow42186)} = 99.90\%$; $\text{PDR}_{(\text{R3}\rightarrow\text{R6}\,:\,42228\rightarrow42166)} = 99.85\%$; $\text{PDR}_{(\text{R3}\rightarrow\text{R7}\,:\,42228\rightarrow42053)} = 99.59\%$; $\text{PDR}_{(\text{R3}\rightarrow\text{R8}\,:\,42228\rightarrow42053)} = 99.59\%$; $\text{PDR}_{(\text{R3}\rightarrow\text{R9}\,:\,42228\rightarrow41986)} = 99.43\%$.



(a) Router R_4 RTT distributions within 3σ.

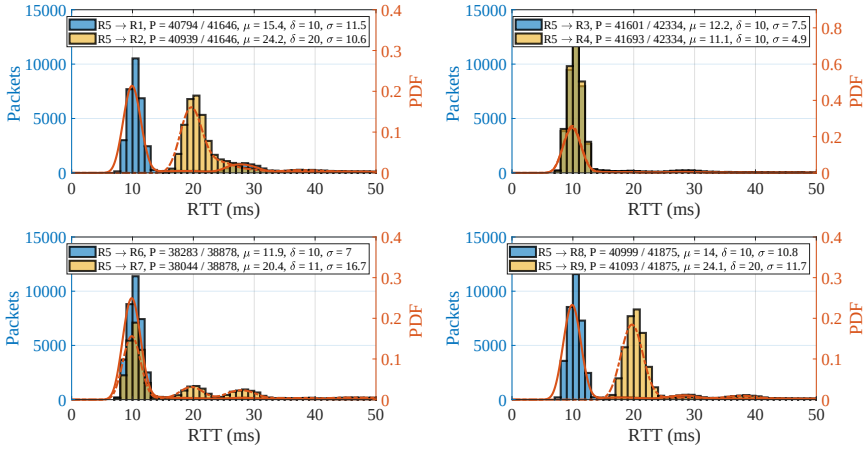


(b) Router R_4 RTT distributions outside 3σ.

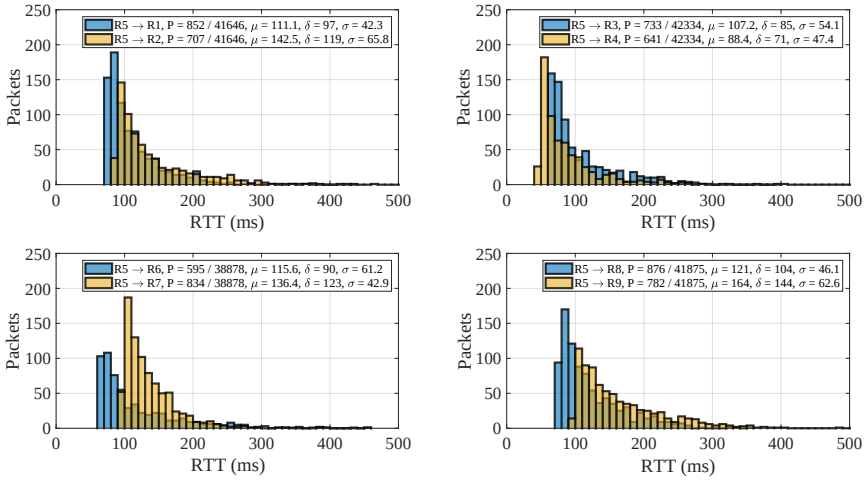


(c) Router R_4 RTT statistics and packet loss.

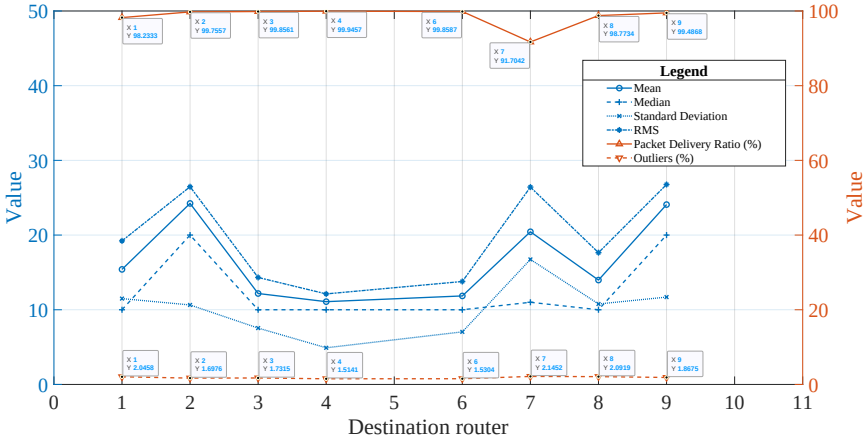
Figure 4.4: Router $R_4 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c)
$PDR_{(R4 \rightarrow R1\,:\,42670 \rightarrow 42641)} = 99.93\%$; $PDR_{(R4 \rightarrow R2\,:\,42670 \rightarrow 42284)} = 99.10\%$;
$PDR_{(R4 \rightarrow R3\,:\,42670 \rightarrow 41578)} = 97.44\%$; $PDR_{(R4 \rightarrow R5\,:\,42670 \rightarrow 42650)} = 99.95\%$;
$PDR_{(R4 \rightarrow R6\,:\,42670 \rightarrow 37104)} = 86.96\%$; $PDR_{(R4 \rightarrow R7\,:\,42670 \rightarrow 42649)} = 99.95\%$;
$PDR_{(R4 \rightarrow R8\,:\,42670 \rightarrow 42489)} = 99.58\%$; $PDR_{(R4 \rightarrow R9\,:\,42670 \rightarrow 42513)} = 99.63\%$.



(a) Router R_5 RTT distributions within 3σ.

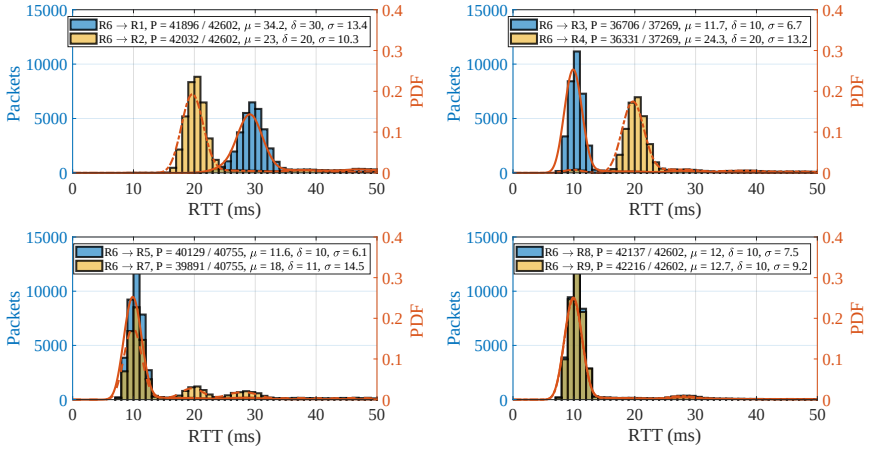


(b) Router R_5 RTT distributions outside 3σ.

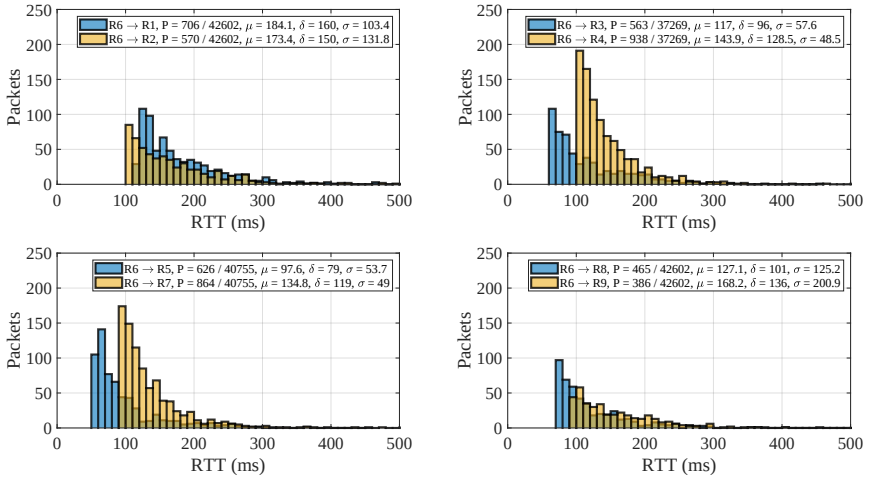


(c) Router R_5 RTT statistics and packet loss.

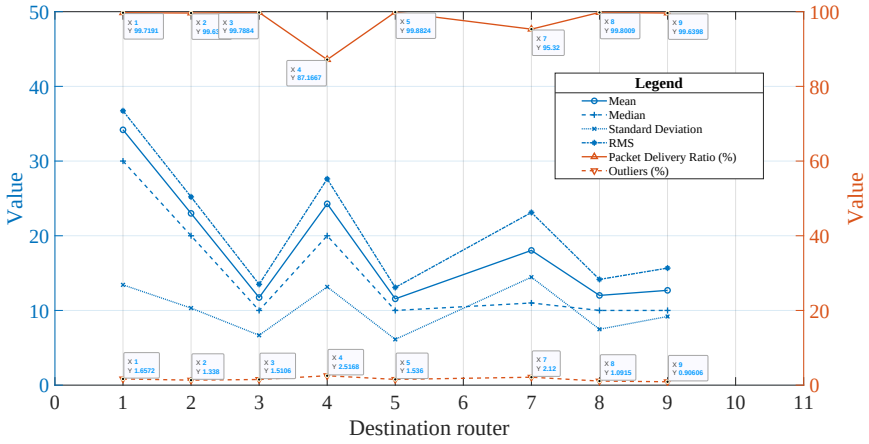
Figure 4.5: Router $R_5 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c)
$\text{PDR}_{(\text{R5}\rightarrow\text{R1}\,:\,42395\rightarrow41646)} = 98.23\%$; $\text{PDR}_{(\text{R5}\rightarrow\text{R2}\,:\,42395\rightarrow42293)} = 99.76\%$;
$\text{PDR}_{(\text{R5}\rightarrow\text{R3}\,:\,42395\rightarrow42334)} = 99.86\%$; $\text{PDR}_{(\text{R5}\rightarrow\text{R4}\,:\,42395\rightarrow42372)} = 99.95\%$;
$\text{PDR}_{(\text{R5}\rightarrow\text{R6}\,:\,42395\rightarrow42340)} = 99.87\%$; $\text{PDR}_{(\text{R5}\rightarrow\text{R7}\,:\,42395\rightarrow38878)} = 91.70\%$;
$\text{PDR}_{(\text{R5}\rightarrow\text{R8}\,:\,42395\rightarrow41875)} = 98.77\%$; $\text{PDR}_{(\text{R5}\rightarrow\text{R9}\,:\,42395\rightarrow42179)} = 99.49\%$.



(a) Router R_6 RTT distributions within 3σ.

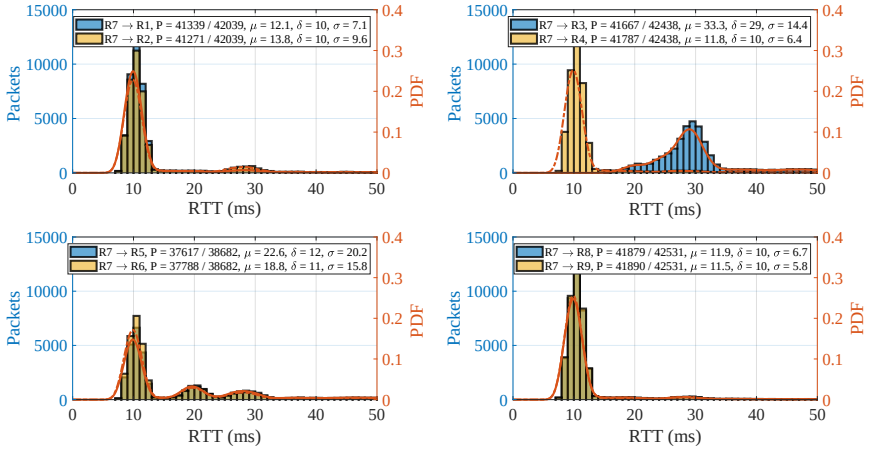


(b) Router R_6 RTT distributions outside 3σ.

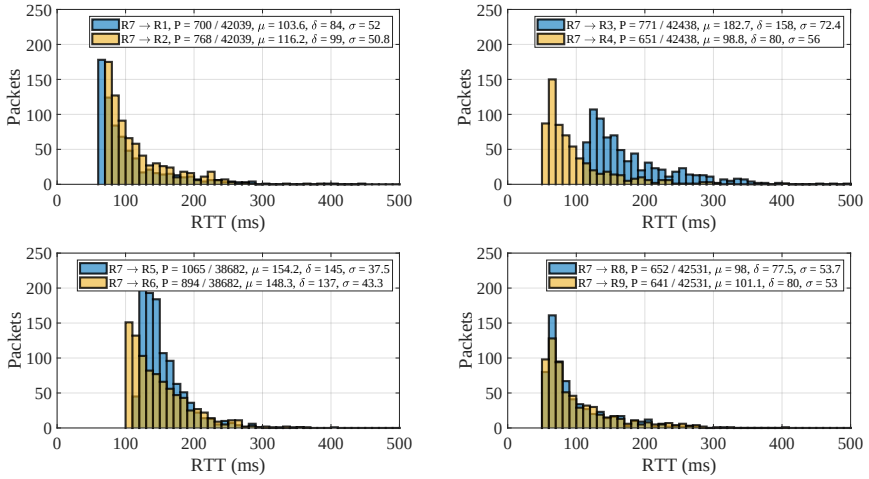


(c) Router R_6 RTT statistics and packet loss.

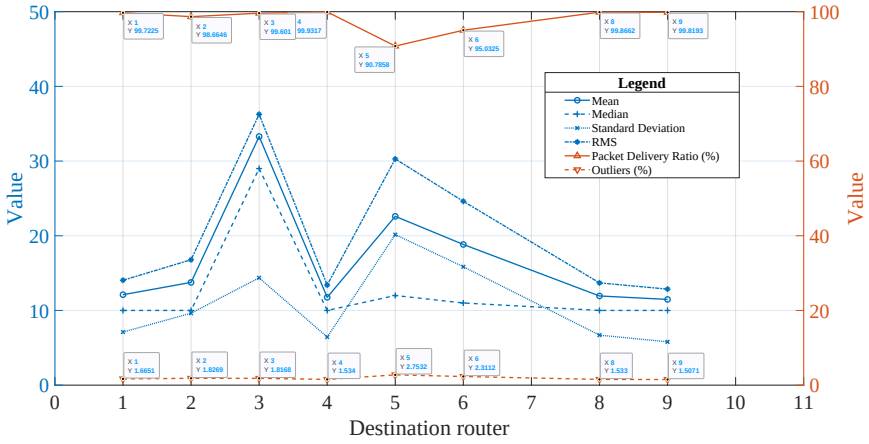
Figure 4.6: Router $R_6 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c)
$PDR_{(R6 \rightarrow R1\,:\,42756 \rightarrow 42636)} = 99.72\%$; $PDR_{(R6 \rightarrow R2\,:\,42756 \rightarrow 42602)} = 99.64\%$;
$PDR_{(R6 \rightarrow R3\,:\,42756 \rightarrow 42677)} = 99.82\%$; $PDR_{(R6 \rightarrow R4\,:\,42756 \rightarrow 37269)} = 87.17\%$;
$PDR_{(R6 \rightarrow R5\,:\,42756 \rightarrow 42708)} = 99.89\%$; $PDR_{(R6 \rightarrow R7\,:\,42756 \rightarrow 40755)} = 95.32\%$;
$PDR_{(R6 \rightarrow R8\,:\,42756 \rightarrow 42671)} = 99.80\%$; $PDR_{(R6 \rightarrow R9\,:\,42756 \rightarrow 42602)} = 99.64\%$.



(a) Router R_7 RTT distributions within 3σ.

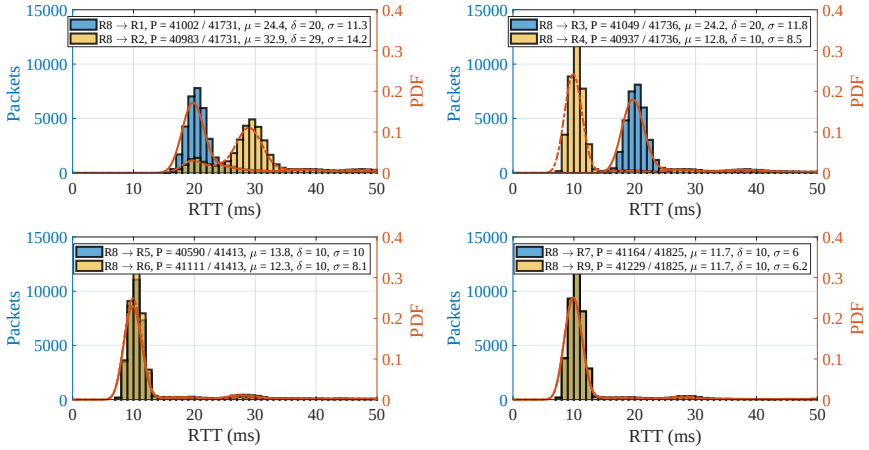


(b) Router R_7 RTT distributions outside 3σ.

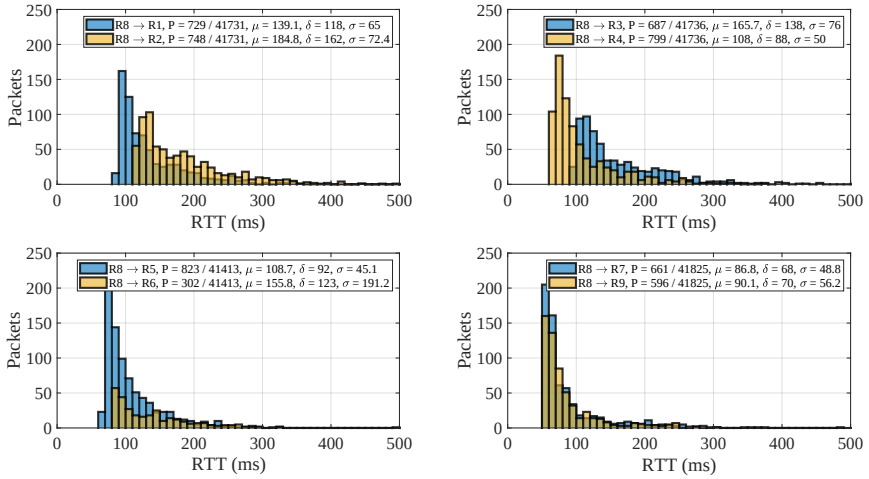


(c) Router R_7 round-trip time statistics and packet loss.

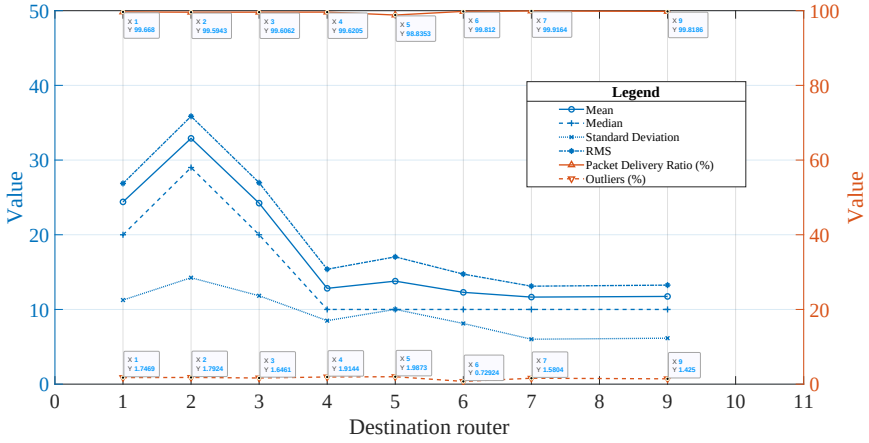
Figure 4.7: Router $R_7 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c)
$PDR_{(R7 \rightarrow R1\,:\,42608 \rightarrow 42491)} = 99.73\%$; $PDR_{(R7 \rightarrow R2\,:\,42608 \rightarrow 42039)} = 98.66\%$;
$PDR_{(R7 \rightarrow R3\,:\,42608 \rightarrow 42438)} = 99.60\%$; $PDR_{(R7 \rightarrow R4\,:\,42608 \rightarrow 42579)} = 99.93\%$;
$PDR_{(R7 \rightarrow R5\,:\,42608 \rightarrow 38682)} = 90.79\%$; $PDR_{(R7 \rightarrow R6\,:\,42608 \rightarrow 40591)} = 95.27\%$;
$PDR_{(R7 \rightarrow R8\,:\,42608 \rightarrow 42551)} = 99.87\%$; $PDR_{(R7 \rightarrow R9\,:\,42608 \rightarrow 42531)} = 99.82\%$.



(a) Router R_8 RTT distributions within 3σ.

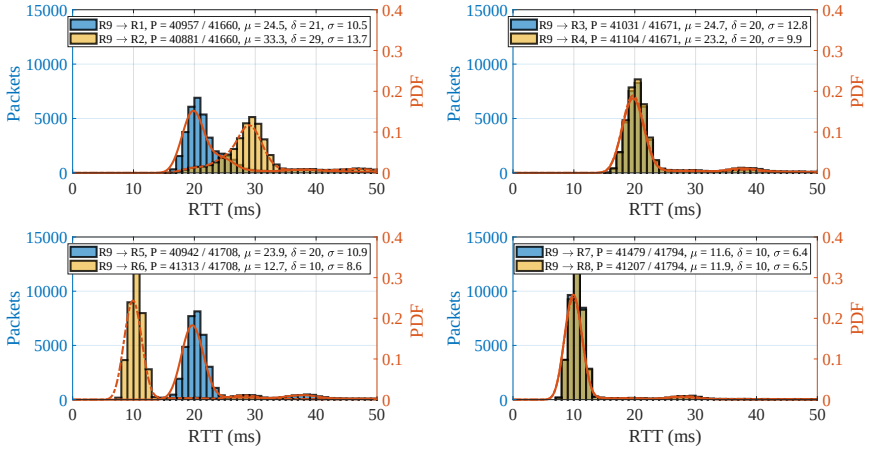


(b) Router R_8 RTT distributions outside 3σ.

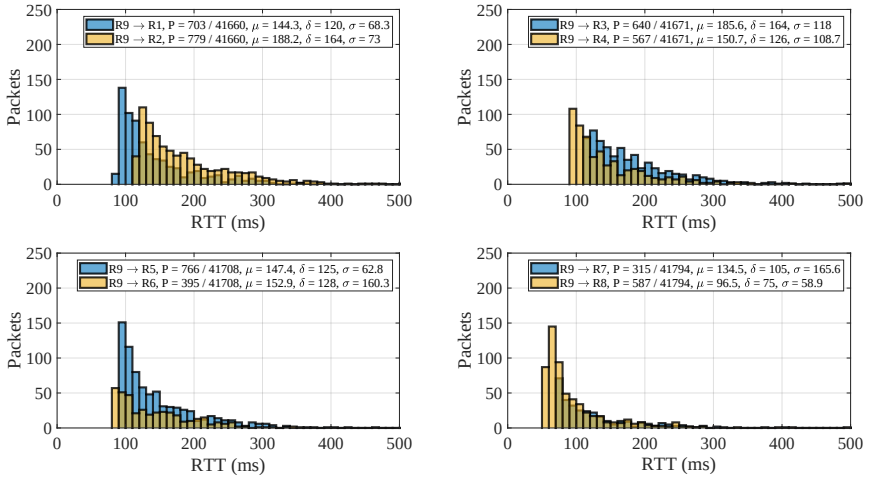


(c) Router R_8 RTT statistics and packet loss.

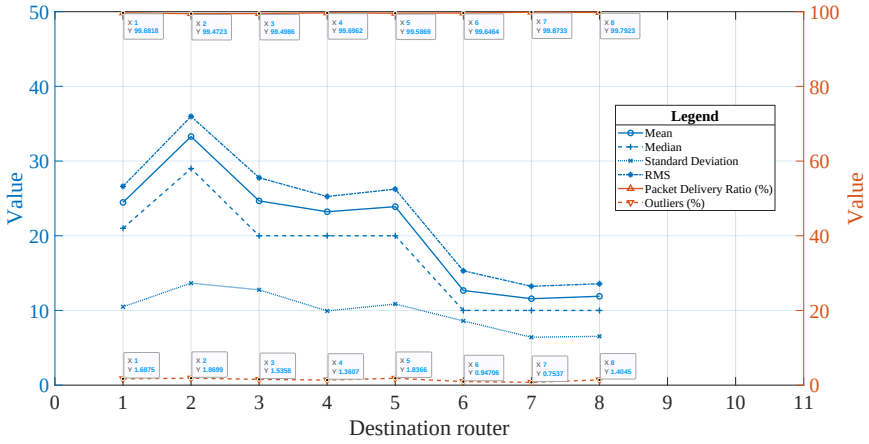
Figure 4.8: Router $R_8 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c)
$PDR_{(R8 \rightarrow R1\,:\,41901 \rightarrow 41762)} = 99.67\%$; $PDR_{(R8 \rightarrow R2\,:\,41901 \rightarrow 41731)} = 99.59\%$;
$PDR_{(R8 \rightarrow R3\,:\,41901 \rightarrow 41736)} = 99.61\%$; $PDR_{(R8 \rightarrow R4\,:\,41901 \rightarrow 41742)} = 99.62\%$;
$PDR_{(R8 \rightarrow R5\,:\,41901 \rightarrow 41413)} = 98.84\%$; $PDR_{(R8 \rightarrow R6\,:\,41901 \rightarrow 41823)} = 98.81\%$;
$PDR_{(R8 \rightarrow R7\,:\,41901 \rightarrow 41866)} = 99.92\%$; $PDR_{(R8 \rightarrow R9\,:\,41901 \rightarrow 41825)} = 99.82\%$.



(a) Router R_9 RTT distributions within 3σ.



(b) Router R_9 RTT distributions outside 3σ.



(c) Router R_9 RTT statistics and packet loss.

Figure 4.9: Router $R_9 \rightarrow R_d$ RTT distributions a) within 3σ b) outside 3σ and c)
$PDR_{(R9 \rightarrow R1\,:\,41881 \rightarrow 41748)} = 99.68\%$; $PDR_{(R9 \rightarrow R2\,:\,41881 \rightarrow 41660)} = 99.47\%$;
$PDR_{(R9 \rightarrow R3\,:\,41881 \rightarrow 41671)} = 99.50\%$; $PDR_{(R9 \rightarrow R4\,:\,41881 \rightarrow 41754)} = 99.70\%$;
$PDR_{(R9 \rightarrow R5\,:\,41881 \rightarrow 41708)} = 99.59\%$; $PDR_{(R9 \rightarrow R6\,:\,41881 \rightarrow 41733)} = 98.65\%$;
$PDR_{(R9 \rightarrow R7\,:\,41881 \rightarrow 41828)} = 99.87\%$; $PDR_{(R9 \rightarrow R8\,:\,41881 \rightarrow 41794)} = 99.79\%$.

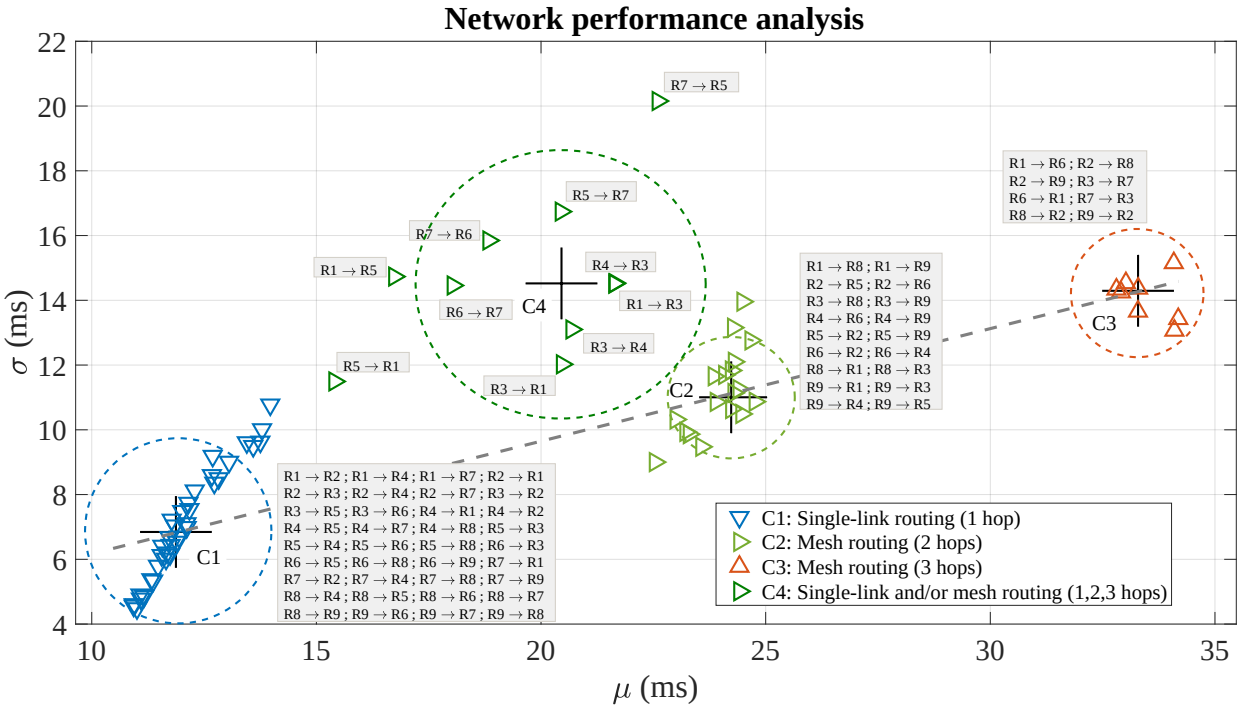


Figure 4.10: Mesh network route performance by *k-means* clustering analysis.

- One-hop packet transmissions have an approximation of $\mu \approx 10$ ms.
- Two-hop packet transmissions have an approximation of $\mu \approx 20$ ms.
- Three-hop packet transmissions have an approximation of $\mu \approx 30$ ms.
- Four-hop packet transmissions have an approximation of $\mu \approx 40$ ms.
- Thread currently allows a maximum of 32 routers in a mesh network with a maximum hop count of 16 hops from router R_s to R_d.
- Round-Trip times are measured and not latency.

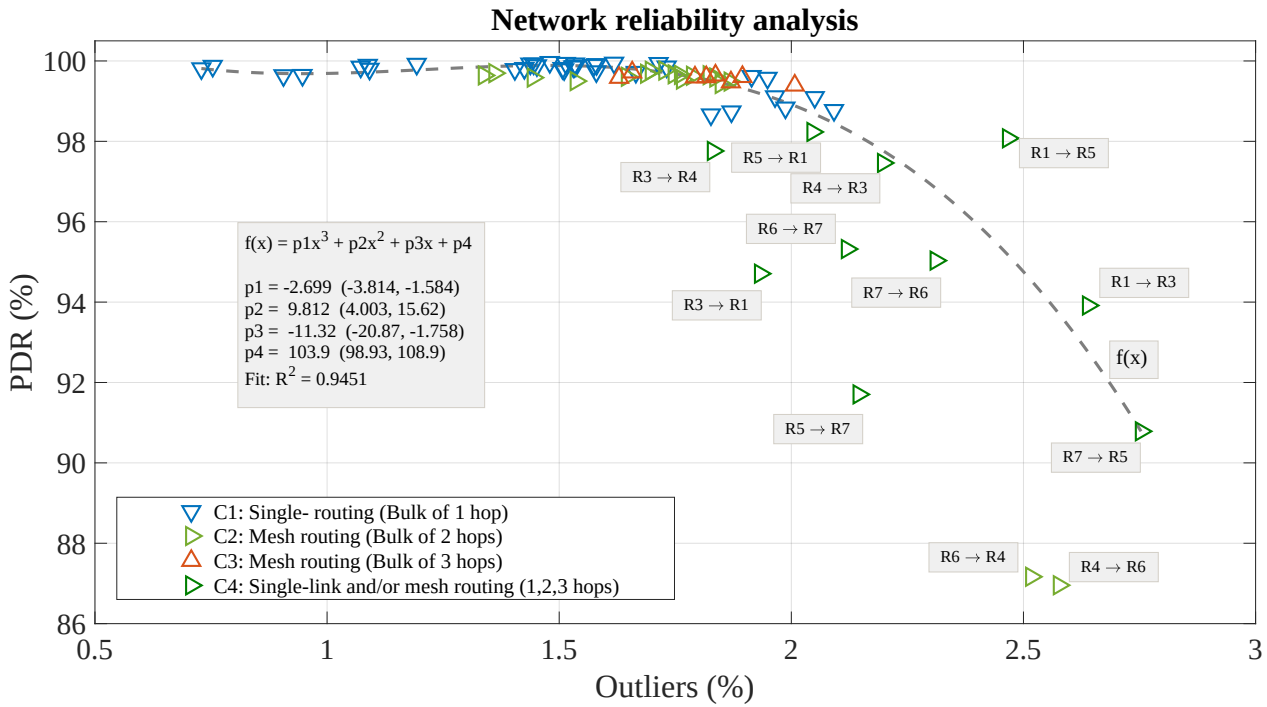


Figure 4.11: Mesh network route reliability by regression analysis.

the RTT distributions are uni-modal and indicate one-hop packet transmissions as shown in **Figure 4.1a**.

ii.) Performance is indicative of a single-link establishment.

2. Multiple-link packet transmissions detected in routes $R_1 \rightarrow \{R_3, R_5, R_6, R_8, R_9\}$:

   i.) $R_1 \rightarrow R_3$: The bulk of RTT distribution is bi-modal with one-hop and two-hop packet transmissions which indicate an active presence of dynamic routing[2] as shown in **Figure 4.1a**. There exist potentially reliability issues, as shown by the *outlier* percentage and *PDR* as shown in **Figure 4.1b** and **Figure 4.1c**.

   ii.) $R_1 \rightarrow R_5$: The bulk of the RTT distribution is uni-modal indicating one-hop packet transmissions. A bump exists in the distribution of three-hop packet transmissions indicating dynamic routing as shown in **Figure 4.1a**. There exist potential reliability issues as shown by the *outlier* percentage and *PDR*, as shown in **Figure 4.1b** and **Figure 4.1c**.

   iii.) $R_1 \rightarrow R_6$: The bulk of the RTT distributions are uni-modal indicating three-hop packet transmissions, as shown in **Figure 4.1a**. No reliability issues present by the *outlier* percentage and *PDR* as shown in **Figure 4.1b** and **Figure 4.1c**.

   iv.) $R_1 \rightarrow \{R_8, R_9\}$: The bulk of the RTT distribution is uni-modal indicating two-hop packet transmissions, as shown in **Figure 4.1a**. No reliability issues present by the *outlier* percentage and *PDR*, as shown in **Figure 4.1b** and **Figure 4.1c**.

As shown in the distribution analysis of route **Figure 4.2** to **Figure 4.9**, the analysis is repeated for source routers R_2 to R_9. Instead of using a summarised approach, the performance metrics computed are used to plot the relationship of the route mean and

[2]Route changes as link cost changes due to noise in the signal strength measurements.

standard deviation. The *k-means* clustering method is employed to group the 72 routes in the mesh and is presented in **Figure 4.10**. From the cluster analysis, the network's performance is summarised as follows:

1. For RTT distributions with values: $\mu \leq 15$ ms; $\delta \leq 10$ ms; $\sigma \leq 11$ ms, 36 routes are clustered around *centroid 1* with the following characteristics:
   i.) Low RTT delays and jitter present in routes.
   ii.) Stable point-to-point with one-hop packet transmissions.
   iii.) Routes don't indicate any reliability issues, more on reliability in **section 4.1.4**.
2. For RTT distributions with values: $\mu \leq 25$ ms; $\delta \leq 20$ ms; $\sigma \leq 17$ ms, 18 routes are clustered around *centroid 2* with the following characteristics:
   i.) Routes have slightly higher RTT delays and jitter compared to routes clustered around *centroid 1*.
   ii.) Mostly mesh with two-hop packet transmissions.
   iii.) Two out of 18 routes indicate reliability issues as shown in **Figure 4.11**, more on reliability in **section 4.1.4**.
3. For RTT route distributions with values: $\mu \leq 34$ ms; $\delta \leq 30$ ms; $\sigma \leq 15$ ms, 8 routes are clustered around *centroid 3*.
   i.) Routes have slightly higher RTT delays and jitter compared to routes clustered around *centroid 2*.
   ii.) Mostly mesh with three-hop packet transmissions.
   iii.) Routes don't indicate any reliability issues as shown in **Figure 4.11**, more on reliability in **section 4.1.4**.
4. For other RTT distributions: 10 routes are clustered around *centroid 4* with the

following characteristics:

i.) A higher variance in RTTs present in routes, data points are more dispersed compared to points around *centroid 1*, *centroid 2* and *centroid 3*.

ii.) Dynamic mesh with one-hop, two-hop and three-hop routing present.

iii.) Routes don't follow a linear relationship compared to the routes clustered around *centroid 1*, *centroid 2* and *centroid 3*.

iv.) Routes indicate reliability issues, more on reliability in **section 4.1.4**.

4.1.4 Reliability analysis

Routes are identified with degraded reliability by studying the *outlier* and *PDR* metrics of $R_s \rightarrow R_d$. As we recall, *outliers* are the percentage of the RTTs that falls outside 3σ and *PDR* indicate as a percentage whether the transmitted packets were successfully delivered. From **Figure 4.11**, the reliability of the routes is computed as:

- $PDR_{(C1)} = 99.7\%$ (36 routes, single link routing of one-hop count).
- $PDR_{(C2)} = 98.2\%$ (18 routes, mesh routing of two-hop counts).
- $PDR_{(C3)} = 99.6\%$ (8 routes, mesh routing of three-hop count).
- $PDR_{(C4)} = 95.3\%$ (10 routes, single link and mesh routing).
- $PDR_{(C1,C2,C3,C4)} = 98.2\%$ (Overall network reliability of 72 routes).

As depicted in **Figure 4.10** and **Figure 4.11**, routes with the poorest reliability are more dispersed around *centroid 4* than other points. From the polynomial model, it can be seen from routes with *outliers* greater than 2%, an exponential degradation in the *PDR* is observed. The routes seem more unstable potentially caused by dynamic routing in the

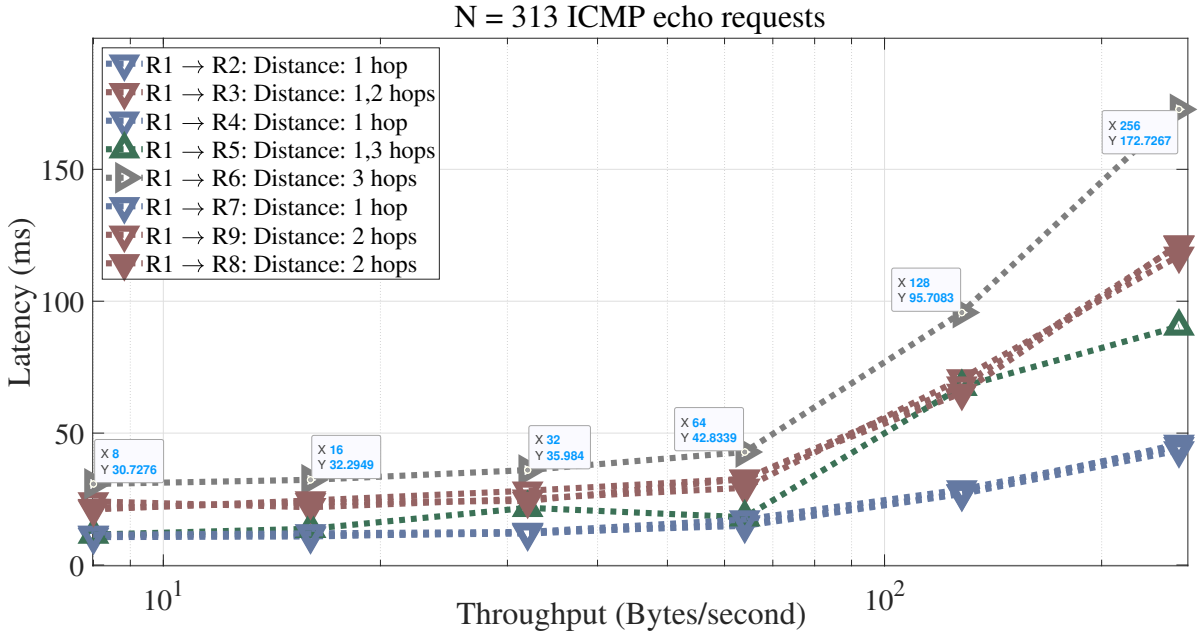


Figure 4.12: Router RTTs at various throughputs.

network. Dynamic routing means that the routing protocol will dynamically change link connections to nearby routers with the least cost because of noise or obstacles in the environment, which may subsequently cause higher jitter and packet losses. Routes with the worst reliability, $R_4 \leftrightarrow R_6$, in both directions through multi-hopping are clustered around *centroid 2* and does not indicate a large extent of dynamic routing (see **Figure 4.4a**) but has the highest packet losses potentially caused by the building's environmental conditions.

4.1.5 Low-throughput versus RTT

In this section, we briefly demonstrate how RTTs are influenced by the payload of a packet. Source router R_1 in **Figure 3.5** is set-up to sequentially sent 313 unicast ICMP requests at different throughputs (8, 16, 24, 32, 64, 128, 256 B/s) to each of the destination routers R_2

to R_9 for analysis. As shown in **Figure 4.12**, the RTT increases for larger packet payloads and the results indicate an acceptable increase in RTTs for a 64-byte packet payload. Therefore, more information may be transmitted between interconnected nodes compared to the 8-byte payload, with similar network performance and reliability expected. RTTs exceeding 200 ms for low-throughput applications will be considered too high since the human response time is in this range.

4.1.6 Routing algorithm simulation

The Thread networking layer uses the Distance Vector Routing (DVR) protocol to relay packets in the mesh. Routers are continually updated by periodically sending routing costs to neighbouring routers by the Trickle algorithm [37, p.113]. DVR applies the simple Bellman-Ford equation [29, p.598] to find the least cost of a link before routing of a packet occurs. The link cost is a metric that forms an integral part of the routing protocol that uses the signal strength of neighbouring routers to determine the shortest distance to a destination router. The router distance is associated with the number of hops a packet needs to travel from a source to destination determined by the DVR algorithm. For example, one-hop means there is a single link directly to the destination router using two routers while two-hops mean there are two links through an intermediate router to the destination router using two routers.

The distribution analysis reveal the hop counts of the transmitted packet by analysing the modalities of the RTT measurements. The statistical results depicted in **Figure 4.10** are confirmed by simulating the Bellman-Ford equation of the physical network. Multiple

signal strength measurements are taken and averaged of connected routers in the network and used as a static metric in the simulation to determine the shortest route from source to destination. For example, with reference to **Figure 3.5** and **Table 4.1**, to find the shortest hop distance between $R_1 \rightarrow R_9$, the Bellman-Ford equation is simulated as follows [29]:

$$D_{(R_1 \rightarrow R_9)} = \min \begin{cases} D_{(R_1 \rightarrow R_9)}, \{c_{(R_1 \rightarrow R_7)} + D_{(R_7 \rightarrow R_9)}\} \\ D_{(R_1 \rightarrow R_9)}, \{c_{(R_1 \rightarrow R_4)} + c_{(R_4 \rightarrow R_7)} + D_{(R_7 \rightarrow R_9)}\} \\ D_{(R_1 \rightarrow R_9)}, \{c_{(R_1 \rightarrow R_5)} + c_{(R_5 \rightarrow R_8)} + D_{(R_8 \rightarrow R_9)}\} \\ D_{(R_1 \rightarrow R_9)}, \{c_{(R_1 \rightarrow R_5)} + c_{(R_5 \rightarrow R_6)} + D_{(R_6 \rightarrow R_9)}\} \\ ..., \end{cases} \tag{4.6}$$

where c is the cost between the intermediary nodes and D is the shortest distance. It is clear from **Equation 4.6** and considering the topology in **Figure 3.5** that the shortest route follows $R_1 \rightarrow R_7$ with a cost of 7, then $R_7 \rightarrow R_9$ with a cost of 8 resulting in a hop distance of 15. Similarly, **Table 4.2** to **Table 4.9** are generated for source routers R_2 to R_9. The router tables confirm the results obtained from the statistical analysis except for $R_1 \leftrightarrow R_3$, $R_1 \leftrightarrow R_5$ in both directions, which are potentially due to fluctuating signal strengths caused by multi-pathing when measurements were taken. Note, in the routing tables that link costs between $R_s \leftrightarrow R_d$ are termed NA, since these routing costs indicate at the time of the RSSI measurements, inconsistent signal strengths in both directions. Reliability degradation is possibly caused by CSMA-CA packet re-transmissions due to the environmental conditions as seen by the *outliers*. Poor link connections cause the DVR

Table 4.1: Router $R_1 \rightarrow R_d$ routing table.

Destination	Cost	Next	Hop
R_1	-	-	-
R_2	3	-	1
R_3	6	R_2	2
R_4	5	-	1
R_5	NA	-	1
R_6	12	R_4	3
R_7	7	-	1
R_8	15	R_7	2
R_9	15	R_7	2

Table 4.2: Router $R_2 \rightarrow R_d$ routing table.

Destination	Cost	Next	Hop
R_1	3	-	1
R_2	-	-	-
R_3	3	-	1
R_4	8	-	1
R_5	7	R_3	2
R_6	NA	R_3	2
R_7	8	-	1
R_8	18	R_1	3
R_9	18	R_1	3

Table 4.3: Router $R_3 \rightarrow R_d$ routing table.

Destination	Cost	Next	Hop
R_1	6	R_2	2
R_2	3	-	1
R_3	-	-	-
R_4	7	R_5	2
R_5	4	-	1
R_6	NA	-	1
R_7	NA	R_2	3
R_8	NA	R_6	2
R_9	NA	R_6	2

Table 4.4: Router $R_4 \rightarrow R_d$ routing table.

Destination	Cost	Next	Hop
R_1	5	-	1
R_2	NA	R_1	1
R_3	7	R_5	2
R_4	-	-	-
R_5	3	-	1
R_6	7	R_5	2
R_7	6	-	1
R_8	10	-	1
R_9	14	R_7	2

Table 4.5: Router $R_5 \rightarrow R_d$ routing table.

Destination	Cost	Next	Hop
R_1	NA	-	1
R_2	7	R_3	2
R_3	4	-	1
R_4	3	-	1
R_5	-	-	-
R_6	4	-	1
R_7	8	-	1
R_8	10	-	1
R_9	10	R_6	2

Table 4.6: Router $R_6 \rightarrow R_d$ routing table.

Destination	Cost	Next	Hop
R_1	12	R_5	3
R_2	NA	R_5	2
R_3	NA	-	1
R_4	NA	R_5	2
R_5	4	-	1
R_6	-	-	-
R_7	NA	-	1
R_8	6	-	1
R_9	8	-	1

Table 4.7: Router $R_7 \rightarrow R_d$ routing table.

Destination	Cost	Next	Hop
R_1	7	-	1
R_2	8	-	1
R_3	13	R_1	3
R_4	6	-	1
R_5	5	-	1
R_6	NA	-	1
R_7	-	-	-
R_8	8	-	1
R_9	8	-	1

Table 4.8: Router $R_8 \rightarrow R_d$ routing table.

Destination	Cost	Next	Hop
R_1	15	R_7	2
R_2	16	R_7	3
R_3	NA	R_6	2
R_4	14	-	1
R_5	10	-	1
R_6	6	-	1
R_7	8	-	1
R_8	-	-	-
R_9	6	-	1

Table 4.9: Router $R_9 \rightarrow R_d$ routing table.

Destination	Cost	Next	Hop
R_1	15	R_7	2
R_2	18	R_7	3
R_3	16	R_6	2
R_4	14	R_7	2
R_5	12	R_6	2
R_6	8	-	1
R_7	8	-	1
R_8	6	-	1
R_9	-	-	-

algorithm to dynamic change link connections in harsh indoor building environments, which may negatively impact performance and reliability in packet transmissions.

4.2 Conclusion

ICMP packet requests were sent between routers of a Thread mesh network, where RTTs and packets loss were logged. Statistical techniques were used by separating the source to destination RTT measurements within 3σ to eliminate *outliers* and to visualise the modalities in the distributions. The results revealed that the distributions in the RTTs were representative of the hop distance a packet travelled between routers and were confirmed by simulating the Bellman-Ford equation used by DVR. Overall, the network provided excellent reliability for single and multiple links in the network. However, reliability degradation in some routes is present, which indicate dynamic routing possibly caused by various factors such obstacle obstructions, incorrect antenna orientation and interference from other 2.4 GHz RF sources.

Further analysis is needed to identify causes of reliability degradation by studying RF propagation using spectrum or link budget analysis, to understand the higher packet losses in some of the routes. More comprehensive link budget analysis will also allow for better network planning in the distribution of more routers in the building area. The routes with the poorest reliability, $R_4 \leftrightarrow R_6, R_5 \leftrightarrow R_7, R_1 \leftrightarrow R_3, R_6 \leftrightarrow R_7, R_3 \leftrightarrow R_4, R_1 \leftrightarrow R_5$, in both T_x and R_x directions, highlighted in **Table 4.1** and **Table 4.9**, are potentially caused by reflections and scattering from obstacles and concrete walls that might have a destructive effect on the signal strength of the receiving routers.

Chapter 5

Discussion and Conclusion

In this dissertation, a mesh network was successfully developed using an IEEE 802.15.4 [19] based wireless technology. The Thread networking protocol was incorporated on the upper layers of the hardware layer of the protocol stack and provided low-power autonomous mesh connectivity of routers in a harsh indoor environment. Thread carries IPv6 natively and is able to easily connect routers to the internet for cloud storage and computing. Hardware and software developments were presented in order to set up a network test-bench, and an algorithm was developed that sent ICMP requests with logging capabilities on each of the routers in the mesh network. The algorithm was converted to source code and flashed to each of the embedded platforms acting as routers to gather data on the performance and reliability of the network. Finally, statistical techniques were used to analyse the data, and a comprehensive overview of the data collected was presented.

5.1 Statement of Objectives

In summary, the following objectives were met:

- The proposition of an anti-theft solution for deployment in educational institutions of South Africa and the industrial automation of intelligent instrumentation systems at the LHC using low-power mesh networks with IoT capabilities.
- Router distribution in the Physics building through the simple log-distance model and average signal strength measurements.
- Development of a mesh network test-bench using nRF52840-DKs routers flashed with the Thread protocol stack.
- Development and implementation of an algorithm with logging capabilities for testing the performance and reliability of transmitting routers deployed in the mesh network.
- Offline analysis of logged data using mainly statistical techniques.

5.2 Conclusion

In **Chapter 1**, a proposition of intelligent instrumentation systems at the LHC and an anti-theft solution for deployment in educational institutions of South Africa using low-power consumption and low-throughput mesh networks with IoT capabilities were introduced. The problem statement, hypothesis, methodology and importance of the research were outlined and research delineations given.

In **Chapter 2**, a technical overview was given of the BLE and Thread protocol stacks where special emphasis was placed on the hardware layer. The PHY and MAC were

presented while illustrating the techniques responsible for the wireless transmission of electromagnetic signals. The literature discussed state-of-the-art and commercially available low-power consumption wireless technologies. BLE is the most widely used low-power consumption wireless technology found in smart-devices today, while Thread natively supports IPv6 and provide autonomous mesh formations with IoT capabilities. Consequently, a network topology was presented in the development of the final conceptual network design for the HTEL anti-theft application. The proposed network topology is depicted in **Figure 2.8** for device communications to a master and **Figure 2.13** for mesh communications to the cloud. The mesh network topology can be used for almost any type of low-power wireless control and monitoring application with the need for IoT connectivity.

In **Chapter 3**, a Thread network was successfully developed in **Figure 3.5** containing nine routers in the Physics building at the University of the Witwatersrand. The average current consumption of the nRF52840 platform acting as a *router* running Thread will generally not exceed an average current of 10 mA and power consumption can significantly be reduced if Thread *end-devices* are used in the network (see **section 3.2.1.3**). Link-budget analysis was partially conducted which resulted in the derivation of a log-distance model in **Figure 3.4** that aided in the distribution of the routers in the building. An algorithm was developed in **Figure 3.6** and flashed to the nRF52840-DK and randomly activated remotely where ICMP requests were sent from each router acting as a transmitter to all the routers in the network acting as receivers. The network test-bench consisted of nine source and eight destination routers and therefore 72 routes for data communications. More than three million ICMP request was transmitted, emulating a busy mesh network between the routes and concurrently logging RTT and packet loss measurements over 65

hours on nine external storage devices connected to the nRF52840-DK. A route is a path a packet travels from the source to destination router given by $R_s \rightarrow R_d$ and have two column vectors with n samples containing the RTT and packet loss measurements.

In **Chapter 4**, statistical techniques were used by separating the source to destination RTT measurements within three-sigma to eliminate *outliers* and to visualise the modalities in the distributions. The results in **Figure 4.10** revealed that the distributions in the RTTs were representative of the hop distance a packet travelled between routers and the results were confirmed by simulating the Bellman-Ford equation presented in **Table 4.1** to **Table 4.9**. Overall, the network provided excellent reliability as illustrated by **Figure 4.11** for single and multiple links in the network. However, reliability degradation in some routes is present, which indicate dynamic routing possibly caused by the networking protocol adapting to the noisy indoor environmental conditions. Further analysis is needed to identify causes of reliability degradation by studying RF propagation using spectrum or more comprehensive link budget analysis, to understand the higher packet losses in some of the routes. The routes with the poorest reliability was presented and concluded that the degradation is potentially caused by reflections and scattering from obstacles and concrete walls that might have a destructive effect on the signal strength of the receiving routers. Furthermore, the RTT increases for larger packet payloads and the results in **Figure 4.12** indicate an acceptable increase in RTTs for a 64-byte packet payload. Therefore, more information may be transmitted between interconnected nodes compared to the 8-byte payload, with similar network performance and reliability expected.

The maximum number of routers allowed in a Thread network is 32 with a maximum hop count of 16. This allows the deployment of the mesh network in large indoor or outdoor

environments which can easily span a heavily obstructed building floor of at least 2000 m^2, where only nine routers were distributed in the physics building. In large buildings with multiple floors, multiple mesh networks can be deployed where each network can be connected to a border router. Border routers can be connected to WiFi or Ethernet networks to communicate with a CMS or network server for supervisory control and data acquisition. Test results in this study revealed that Thread had provided total route reliability of 98.2% and proven a reliable low-power mesh networking protocol for low-throughput data communications. Application deployment of the Thread wireless technology is feasible for low-throughput monitoring and control applications in harsh indoor environments or educational institutions that have a minimal network infrastructure that requires IoT connectivity.

5.3 Discussion

Since the inception of the HTEL initiative in researching low-power wireless technologies, it is evident that the connectivity of 'small' intelligent devices to the internet will play a massive role in technological developments. Innovative applications using low-power mesh networks for monitoring and control, to advance decision making from intelligent devices communicating to cloud computing services will provide a cost-effective solution for organisations to make informed decisions and optimise operational processes.

IoT solutions currently developed at CERN employ LoRaWAN which is a commercial available sub-GHz long-range and low-power wireless technology using a star-of-stars network topology. Battery-powered intelligent devices are connected to a central network

server through a gateway over the internet. LoRaWAN offer free membership to the 'LoRa Alliance' for researchers who are interested in adopting the wireless technology for their applications. In order to market products that uses LoRaWAN, a 'contributor' membership is required and renewed annually in order to certify LoRaWAN products.[1]

Similarly, the University of the Witwatersrand is an active member of the 'ThreadGroup' and exempted in paying membership fees to research the Thread wireless technology. Thread allows deployment of multiple mesh network connected through gateways for monitoring and control of instrumentation systems in different environments. Once an HTEL application passed the research phase and the commercialisation phase commence, an 'implementer' membership is required to certify products which adopt the Thread technology and bring final products to market. The membership requires a fee (typically with no royalty payments) in order to certify Thread products for continued marketing and is payable to 'ThreadGroup' annually.[2]

5.4 Future Work

A feasibility study will be required before potential commercialisation of the proposed HTEL anti-theft system. Once found feasible, the application will focus on proprietary firmware and Android application development with a reputable tablet manufacturer to render the device inoperable. More broadly speaking, the network development conducted in this research can be used for almost any type of instrumentation system that requires

[1] https://lora-alliance.org/member-directory.
[2] https://www.threadgroup.org/thread-group#OurMembers.

low-throughput wireless data communications and IoT connectivity to a cloud storage or computing service. Data communications of devices to the network will be monitored and controlled from a CMS, and the project may further be extended into other research areas in physics or electrical engineering that involve:

- The study of RF propagation and attenuation in harsh building environments by modelling and more comprehensive link budget analysis.
- Online data analysis and anomaly detection of the mesh networks by the CMS.
- Indoor localisation or tracking of devices by the CMS.

The HTEL will focus on a PCB design embedding the nRF52840-SoC using the guidelines stipulated in **Appendix II**. An external programmer will be used to program the SoC thereby omitting the MCU labelled U2 in **Figure 3.2**. A commercial Altium licence and vector network analyser will be acquired for the PCB and RF developments.

References

[1] IoT at the LHC, 2017. *Integrating Internet-of-Things devices into the control systems for the Large Hadron Collider.* [ONLINE] Available at https://videos.cern.ch/record/2290435 [Accessed 27 December 2019].

[2] Evans, L., Bryant, P., 2008. *LHC machine.* Journal of Instrumentation, Volume 3. doi: 10.1088/1748-0221/3/08/S08001.

[3] Rossi, L., 2003. *The LHC main dipoles and quadrupoles toward series production.* in IEEE Transactions on Applied Superconductivity, vol. 13, no. 2, pp. 1221-1228, June 2003. doi: 10.1109/TASC.2003.812639.

[4] Sonnemann, 2001. *Resistive transition and protection of LHC superconducting cables and magnets.* Doctor of Science Dissertation, RWTH Aachen University, Germany.

[5] Denz, R., 2006. *Electronic systems for the protection of superconducting elements in the LHC.* in IEEE Transactions on Applied Superconductivity, vol. 16, no. 2, pp. 1725-1728, June 2006. doi: 10.1109/TASC.2005.864258.

[6] Fernandez, A.V., Rodriguez-Mateos, F., 2004. *Reliability of the quench protection system for the LHC superconducting elements.* Journal of Nucl. Instrum. Methods Phys. Res., A, Volume 525, p.439-446. doi: 10.1016/j.nima.2004.01.081.

[7] Buchanan W.J., 2004 *WorldFIP: The Handbook of Data Communications and Networks.* (Springer, Boston, MA), ISBN: 978-1-4757-1067-0.

[8] Wielgosz, M., Skoczeń, A., De Matteis, E., 2018. *Protection of superconducting industrial machinery using RNN-based anomaly detection for implementation in smart sensor.* in Sensors (Basel, Switzerland) vol. 18,11 3933. 14 Nov. 2018. doi:10.3390/s18113933.

[9] LoRaWAN at CERN, 2019. *Put a sensor in your life with the new LoRaWAN network.* [ONLINE] Available at https://home.cern/news/announcement/computing/put-sensor-your-life-new-lorawan-network [Accessed 27 December 2019].

[10] Sundaram, J., Du, W., Zhao, Z., 2019. *A survey on LoRa networking: Research problems, current solutions and open issues.* in IEEE Communications Surveys and Tutorials. doi: 10.1109/TASC.2003.812639.

[11] The Citizen, 2015. *Schools tablet theft shock.* [ONLINE] Available at https://citizen.co.za/news/south-africa/382489/schools-tablet-theft-shock/ [Accessed 12 April 2017].

[12] BusinessTech, 2015. *Gauteng withdraws 88,000 tablets from schools because of theft.* [ONLINE] Available at https://businesstech.co.za/news/government/87334/gauteng-withdraws-88000-tablets-from-schools-because-of-theft/ [Accessed 13 April 2017].

[13] Wits Educational Technology, 2017. *Consumer electronics theft in South Africa - A new approach to prevent massive theft of electronics.* Unpublished investigation report by an independent consultant. Report available at the Electronics Laboratory, Room P22A, Physics Building, University of the Witwatersrand.

[14] iOS Security, 2019. *Find my iPhone activation lock.* [ONLINE] Available at https://support.apple.com/en-za/HT201365 [Accessed 14 March 2019].

[15] iOS Security, 2018. *iOS 12.1 security guide.* [ONLINE] Available at https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf [Accessed 18 January 2019].

[16] iOS Security, 2019. *If you forgot the pass-code for your iPhone, iPad, or iPod touch, or your device is disabled.* [ONLINE] Available at https://support.apple.com/en-us/HT204306 [Accessed 14 March 2019].

[17] Absolute, 2018. *Theft recovery solutions.* [ONLINE] Available at https://homeoffice.absolute.com/ [Accessed 29 October 2018].

[18] Absolute, 2019. *Application persistence.* [ONLINE] Available at https://www.absolute.com/en/resources/faq/application-persistence [Accessed 20 March 2019].

[19] IEEE 802.15.4, 2016. *IEEE standard for low-rate wireless networks.* IEEE std 802.15.4-2015 (Revision of IEEE std 802.15.4-2011) , vol., no., pp.1-709, 22 April 2016. doi: 10.1109/IEEESTD.2016.7460875

[20] van Rensburg, R., 2017. *System control applications of low-power frequency devices.* Journal of Physics: Conference Series, Volume 889, conference 1. doi: 10.1088/1742-6596/889/1/012016.

[21] Nordic Semiconductor Report, 2019. *Nordic Semiconductor Q4 and preliminary full year report.* [ONLINE] Available at https://www.nordicsemi.com/-/media/Investor-Relations-and-QA/Quartely-Reports/2019/Q4_Quarterly_report_2019.pdf?la=en&hash=9565BC40E259AEB5100D8BBE5C79721F518CF4DA [Accessed 21 March 2020].

[22] Linear Technology, 2019. *Ultra-low voltage step-up converter and power manager.* [ONLINE] Available at https://www.analog.com/en/products/ltc3108.html [Accessed 8 April 2019].

[23] Maxim Integrated, 2019. *Energy-harvesting charger and protector.* [ONLINE] Available at https://www.maximintegrated.com/en/products/power/battery-management/MAX17710.html [Accessed 8 April 2019].

[24] Online Power Profiler, 2019. *Nordic Semiconductor BLE online power profiler.* [ONLINE] Available at https://devzone.nordicsemi.com/power/ [Accessed 11 April 2019].

[25] van Rensburg, R., 2017. *nRF52 'online power profiler' battery life calculation check.* [ONLINE] Available at https://devzone.nordicsemi.com/f/nordic-q-a/19081/nrf52-online-power-profiler-battery-life-calculation-check [Accessed 12 April 2019].

[26] van Rensburg, R.M., Mellado, B., Augusto Marin Tobon, C., (2019). *Wireless mesh network data communications and reliability analysis for anti-theft application deployment in educational institutions.* Journal of Engineering, Design and Technology, Volume 17 Number 3, p.474-489. doi: 10.1108/JEDT-08-2018-0138.

[27] IEEE Std 802.15.1, 2005. *IEEE Standard for information technology - Local and metropolitan area networks - Specific requirements - Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPAN).* IEEE Std 802.15.1-2005 (Revision of IEEE std 802.15.1-2002), vol., no., pp.1-700, 14 June 2005. doi: 10.1109/IEEESTD.2005.96290.

[28] Burbank, J., Andrusenko, J., Everett, J., Kasch, W., 2013. *Wireless Networking: Understanding internetworking challenges.* 1st edition (New Jersey: John Wiley & Sons, Inc.). ISBN: 978-1-118-12238-9.

[29] Forouzan, B., 2013. *Data communication and networking.* 5th edition (New York: McGraw Hill). ISBN: 978-0-07-337622-6.

[30] Townsend, K., Cufi, C., Davidson, R., 2014. *Getting started with Bluetooth low energy: Tools and techniques for low-power networking.* 1st edition (California: O'Reilly Media). ISBN: 978-1-491-94951-1.

[31] Bluetooth Core Specification v5.1, 2019. *Specification of the Bluetooth system.* [ONLINE] Available at

https://https://www.bluetooth.com/specifications/bluetooth-core-specification/ [Accessed 28 June 2019]

[32] Xiong, F., 2006. *Digital modulation techniques.* 2nd edition (Boston: Artech House). ISBN: 1-58053-863-0. ISBN: 1-58053-863-0.

[33] Lacaita, A., Levantino, S., Samori C., 2007. *Integrated frequency synthesizers for wireless systems.* 1st ed (New York: Cambridge University Press). ISBN: 978-0-521-86315-5.

[34] Su, P., Pamarti, S., 2010. *Fractional-N phase-locked-loop based frequency synthesis: A tutorial.* in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 56, no. 12, pp. 881-885, Dec. 2009. doi: 10.1109/TCSII.2009.2035258.

[35] Heydon, R., 2013. *Bluetooth-low energy: The developers handbook.* 1st edition (Indiana: Prentice Hall). ISBN-13: 978-0-13-288836-3.

[36] Bluetooth Mesh, 2019. *Bluetooth Mesh: A scalable mesh technology with Nordic Semiconductor.* [ONLINE] Available at https://www.nordicsemi.com/Products/Low-power-short-range-wireless/Bluetooth-mesh [Accessed 23 March 2020].

[37] Thread Group, 2017. *Thread specification v1.1.* [ONLINE] Available at https://www.threadgroup.org/ThreadSpec/ [Accessed 23 February 2017].

[38] Thread Group Members, 2019. *University of the Witwatersrand Thread Group research membership.* [ONLINE] Available at https://www.threadgroup.org/thread-group [Accessed 23 March 2020].

[39] MathWorks Communications Toolbox Documentation, 2019. *Link budget analysis.* [ONLINE] Available at [Accessed 2 August 2019].

[40] Nalineswari, D., Rakesh, N., 2015. *Link budget analysis on various terrains using IEEE 802.16 WIMAX standard for 3.5 GHz frequency.* in IEEE International

Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, 2015, pp. 1-5. doi: 10.1109/ICECCT.2015.7226096.

[41] Rouphael, T., 2009. *RF and digital signal processing for software-defined radio.* 1st edition (Burlington: Newnes - Elsevier). ISBN: 978-0-7506-8210-7.

[42] Levis, P., Clausen, T., Hui, J., Gnawali, O., Ko, J.,"RFC 6206: *The Trickle Algorithm*," 4 March 2011. [ONLINE] Available at https://tools.ietf.org/html/rfc6206 [Accessed 20 June 2018].

[43] Wits Thread Membership, 2018. *University of the Witwatersrand Thread research membership benefits.* [ONLINE] Available at https://www.threadgroup.org/thread-group#OurMembers [Accessed 16 August 2018].

[44] Roy, S., et al., 2018. *The effect of locations on the 3-D printing bed surface for designing PC-ABS based RF circuits.* in IEEE International Symposium on Antennas and Propagation and USNC/URSI National Radio Science Meeting, pp. 1437-1438, 2018, doi: 10.1109/APUSNCURSINRSM.2018.8608234.

[45] nRF52840 Product Specification, 2018. *Nordic Semiconductor nRF52840 objective product specification version 1.0.* [ONLINE] Available at https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.0.pdf [Accessed 8 August 2018].

[46] CoreMark, 2020. *EEMBC: EDN Embedded Microprocessor Benchmark Consortium.* [ONLINE] Available at https://www.eembc.org/coremark/scores.php [Accessed 17 June 2020].

[47] nRF52840 Product Specification, 2019. *Nordic Semiconductor nRF52840 objective product specification version 1.1.* [ONLINE] Available at https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf [Accessed 15 September 2019].

[48] nRF52840-DK PCA10056 v1.0.0, 2018. *nRF52840 development kit PCA10056 version 1.0.0 user guide version 1.2.* [ONLINE] Available at https://www.nordicsemi.com/-/media/DocLib/Other/User_Guides/nRF52840DKUserGuidev12pdf.pdf [Accessed 11 November 2018].

[49] Nordic Semiconductor Infocenter, 2018. *nWP-017: Antenna tuning design note v1.0.* [ONLINE] Available at http://infocenter.nordicsemi.com/pdf/nwp_017.pdf [Accessed 31 October 2018].

[50] Mindan, B., Hong, L., 2010. *The analysis of impedance matching problem in RF circuit design.* Forum on Information Technology and Applications, Kunming, 2010, pp. 350-353. doi: 10.1109/IFITA.2010.55.

[51] OpenThread, 2017. *nRF52 SDK for Thread version 0.10.0.* [ONLINE] Available at https://github.com/openthread/openthread/ [Accessed 1 February 2017].

[52] GNU Arm Embedded Toolchain, 2019. *GNU Arm Embedded Toolchain: Pre-built GNU bare-metal toolchain for 32-bit Arm processors.* [ONLINE] Available at https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm [Accessed 23 July 2020].

[53] GCC, 2019. *GCC, the GNU Compiler Collection.* [ONLINE] Available at https://gcc.gnu.org/ [Accessed 24 July 2020].

[54] Segger, 2019. *Segger: J-Link Debug Probes.* [ONLINE] Available at https://www.segger.com/products/debug-probes/j-link/ [Accessed 24 July 2020].

[55] Tadiran Batteries, 2018. *Tadiram non-rechargeable 35 Ah/3.6 V lithium-thionyl chloride batteries.* [ONLINE] Available at http://www.tadiranbat.com/assets/tl-5937.pdf [Accessed 17 December 2019].

[56] Rappaport, T., 2002. *Wireless communications principles and practice.* 2nd ed (New York: Prentice Hall). ISBN: 9780130422323.

[57] van Rensburg, R.M., Mellado, B., Sandrock, C., 2018. *System control applications of low-power frequency devices.* in Proceedings of SAIP2017, the 62nd Annual Conference of the South African Institute of Physics, edited by Prof. Japie Engelbrecht (SU/2017), pp. 373 - 378. ISBN: 978-0-620-82077-6. Available online at http://events.saip.org.za/getFile.py/access?resId=64&materialId=8&confId=91

[58] van Rensburg, R., 2019. *Wireless mesh network ICMP data.* [ONLINE] Available at https://www.researchgate.net/publication/330824477_Mesh_network_ICMP_data doi: 10.13140/RG.2.2.32648.88322

[59] Nordic Semiconductor Infocenter, 2017. *nRF52840 SoC schematic and printed circuit board reference design guidelines.* [ONLINE] Available at https://infocenter.nordicsemi.com/pdf/nRF52840_OPS_v0.5.pdf [Accessed 5 May 2018].

APPENDICES

The appendices are outlined as follows:

Appendix I:
- PCD anti-theft solutions.

Appendix II:
- nRF52840 PCB design guidelines.

Appendix III:
- ICMP algorithm and logging C code.

Appendix IV:
- Gaussian kernel Matlab script.

Appendix I:

PCD anti-theft solutions

The following advantages and disadvantages found in this study are outlined for devices operating on iOS, Android and third-party security firmware. In summary, iOS offers a proprietary operating system to their devices while Android's operating system is open access where security firmware from third-party companies may be activated. Alternatively, Android provides open-source ROMs developed by companies or community of developers where firmware can be changed. However, Android firmware can be left unchanged and only third-party firmware may be activated on the PCD to protect the Android application and preventing flashing of the ROM[1].

Advantages of the iOS operating system:

1. Device ROM cannot be flashed and replaced by customised ROMs that may compromise iOS security features.
2. Tracking, device-lock and data-deletion capabilities from a remote location if a lost or stolen is powered on and connected to the internet. Requires a computer to connect to the iCloud for security features to work.
3. Third-party APPs available for tracking.
4. When Activation Lock is enabled, the device is rendered useless without the correct Apple username and password. This security feature is an extremely strong deterrent of tablet theft making organised crime more difficult.

Disadvantages of the iOS operating system:

1. Devices running the iOS operating system are generally more expensive compared to Android devices. Unsuitable for teaching and learning in educational institutions in the South African context.
2. Devices run on proprietary operating systems. Any upgrades or modifications to the firmware is done by Apple over the iCloud.

Advantages of the Android operating system:

1. More cost-effective compared to devices running proprietary operating systems.

[1]ROM in tablets is a flashable non-volatile memory that contains machine code from an image file of the Android applications, operating system and firmware. Bricked means the PCD operating system operability is inaccessible rendering a device useless.

2. More suited for teaching and learning in South African educational institutions.
3. Most popular operating system used in the world for smart devices.
4. Remote tracking capabilities, device-lock and data-deletion if a lost or stolen device is powered on and connected to the internet.
5. Device ROM may be flashed using customised ROMs which also allows Android developers to develop their own operating systems.
6. Third-party security firmware embedded in most processors operating on Android.
7. Third-party APPs available for tracking.

Disadvantages of the Android operating system:

1. Requires a PC to connect to the internet for tracking.
2. Requires technical expertise in Android firmware to upgrade or customise ROMs.
3. Device can be easily reset to factory settings when lost or stolen.
4. The activation of third-party security firmware is expensive and requires a lost or stolen device a connection to the internet to work.

Advantages of the Persistence firmware:

1. Security firmware already embedded on major chip manufacturers.
2. Tracking capabilities if the lost or stolen device is powered on and connected to the internet. Private investigations teams are provided to retrieve stolen devices.
3. Prevents flashing of ROM and deletion of installed applications.

Disadvantages of the Persistence firmware:

1. Requires an annual subscription.[2]
2. Requires a PC to connect to the web for tracking.
3. Requires internet for Persistence to work.
4. May be extremely expensive to activate for thousands of devices.

Note: The Persistence firmware is not embedded on the SoC BIOS but on the Solid State Drive (SSD). Once the SSD is formatted then all the Persistence and tracking features are lost.

[2] https://homeoffice.absolute.com/product-comparison/.

Appendix II:

nRF52840 PCB design guidelines

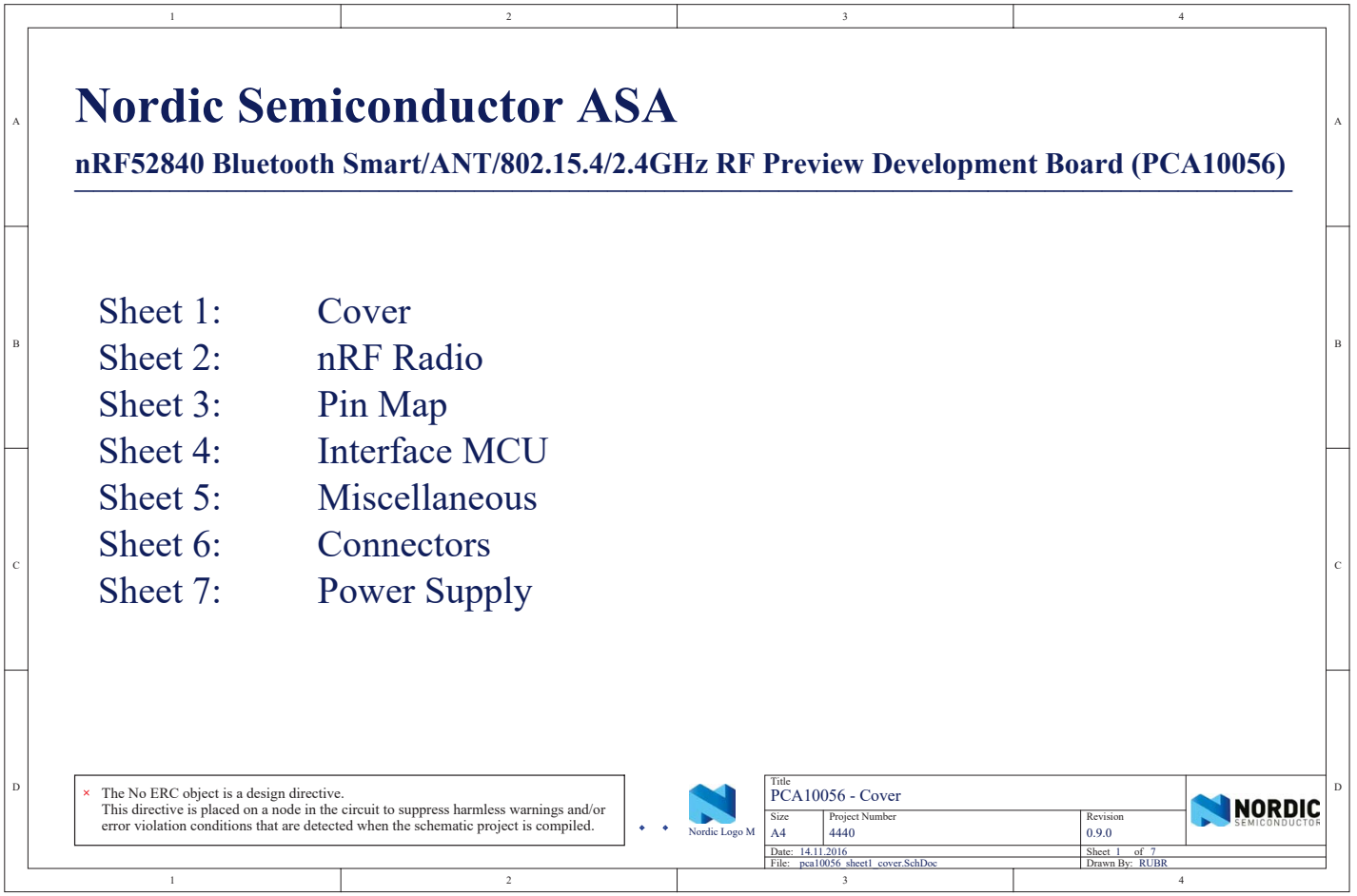
Nordic Semiconductor recommends PCB layout designs embedding the nRF52840-QIAA in **SCH II.2** with the values of the components in table **BOM II.1** to achieve similar RF performance as the nRF52840-DK depicted in table **PCB II.1** to **PCB II.4**.

General guidance is summarised in nRF52840 Specification v1.1 [47, p.61,80,583]:

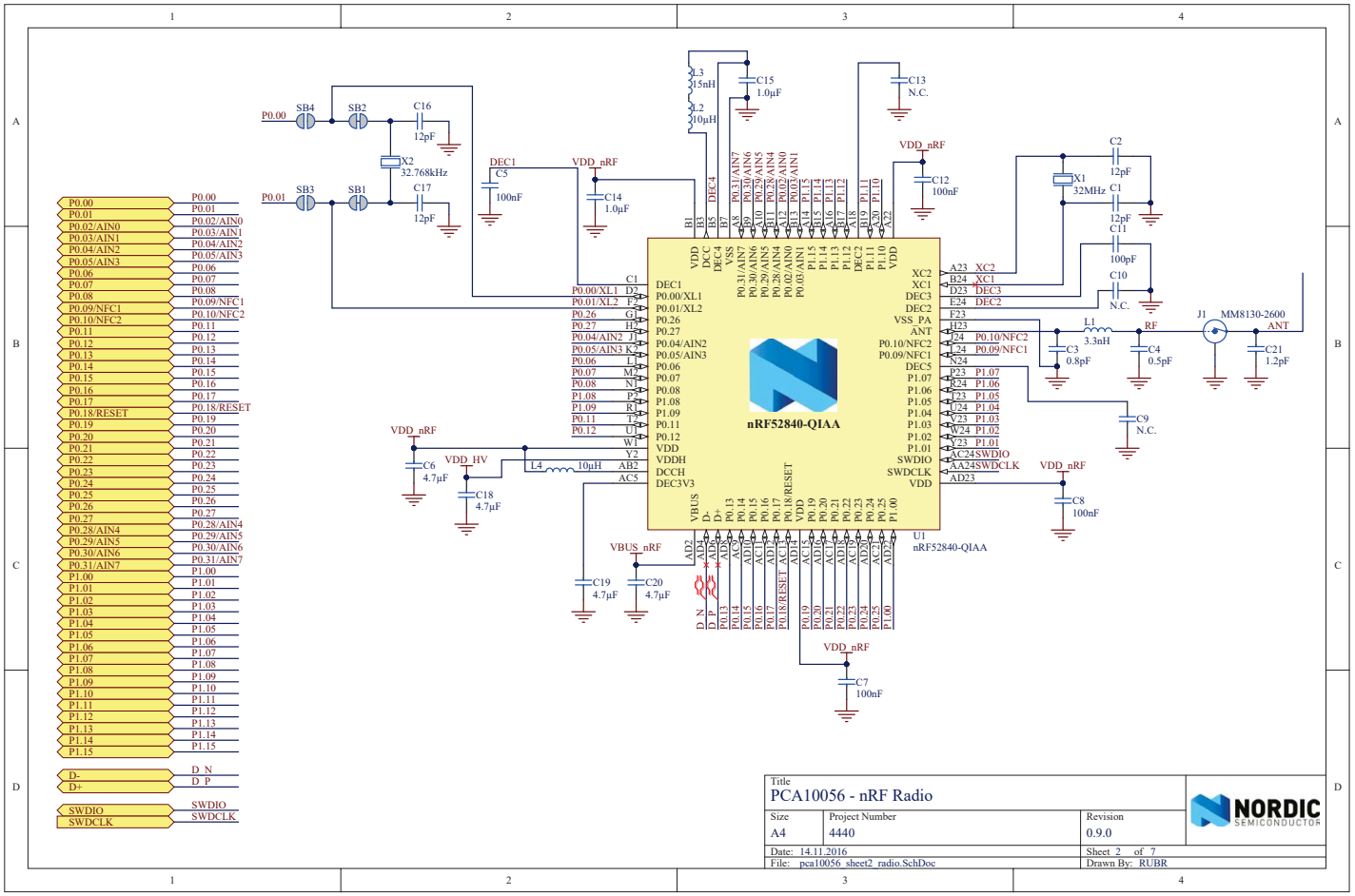
1. "Normal voltage mode is entered when the supply voltage is connected to both the VDD and VDDH pins (pin VDD shorted to pin VDDH)." "The external supply is annotated with the VEXT net name." The nRF52840-QIAA provides a DC/DC buck and LDO regulator. VDD is typically 3.0 V.
2. "When supplying power from a USB source only, VBUS must be connected to VDDH if USB is to be used."

General design guidelines is summarised in nRF52840 Specification v1.1 [47, p.608-610]:

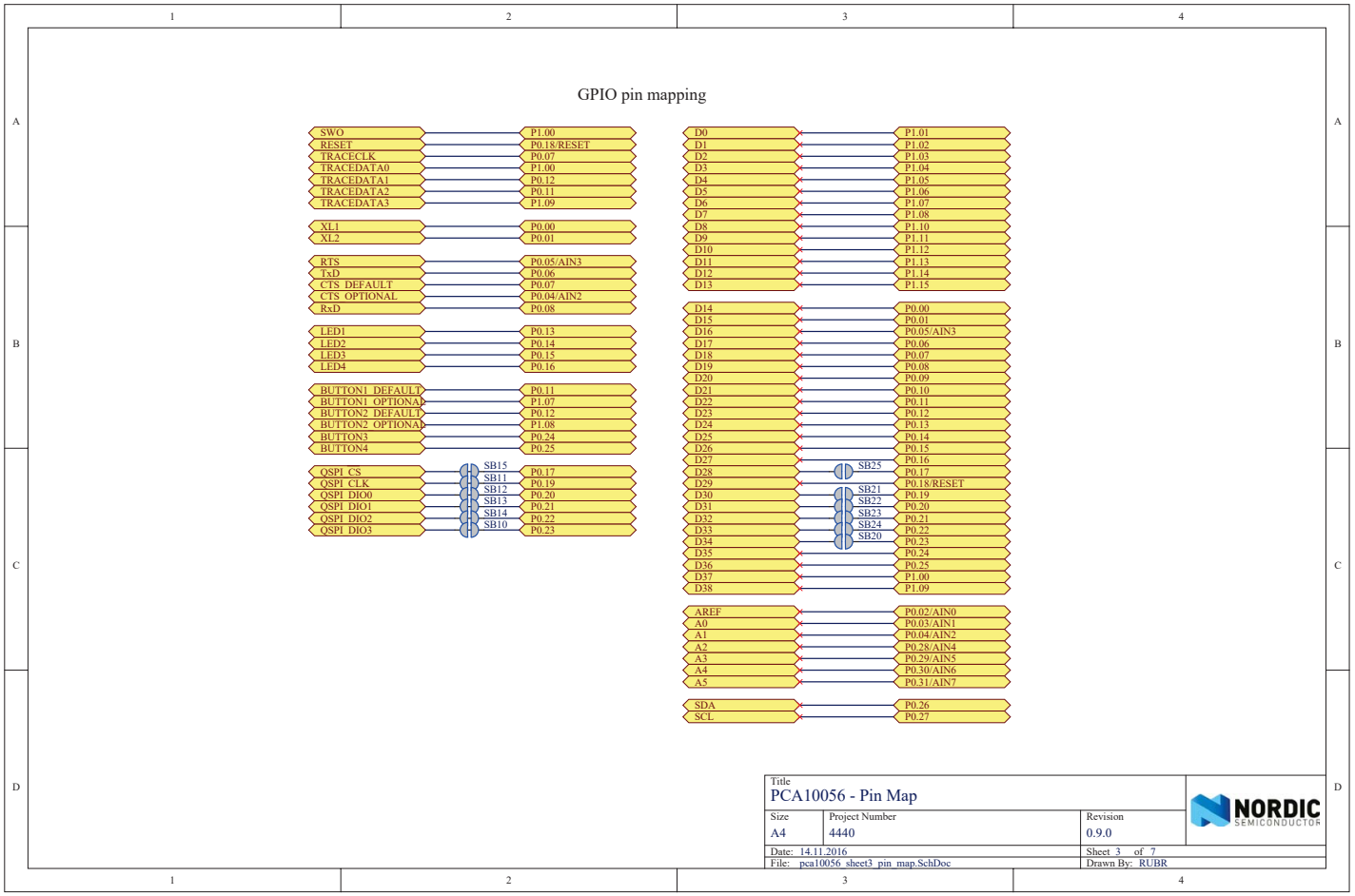
1. "A PCB with a minimum of two layers, including a ground plane, is recommended for optimal RF performance." "On PCBs with more than two layers, put a keep-out area on the inner layers directly below the antenna matching circuitry to reduce the stray capacitances that influence RF performance."
2. "A matching network is needed between the RF pin ANT and the antenna, to match the antenna impedance of normally 50-ohm to the optimum RF load impedance."
3. "The DC supply voltage should be decoupled as close as possible to the VDD pins with high-performance RF capacitors." "The supply voltage for the chip should be filtered and routed separately from the supply voltages of any digital circuitry."
4. "Long power supply lines on the PCB should be avoided." "All device grounds, VDD connections, and VDD bypass capacitors must be connected as close as possible to the IC." "For a PCB with topside RF ground plane, the VSS pins should be connected directly to the ground plane."
5. "Fast switching digital signals should not be routed close to the crystal or the power supply lines."
6. "Capacitive loading of fast switching digital output lines should be minimised to avoid radio interference."
7. The minimum PCB layout requirements (without a programmer) of the nRF52840 aQFN-73 package is shown in nRF52840 Specification v1.1 [47, p.609-610].



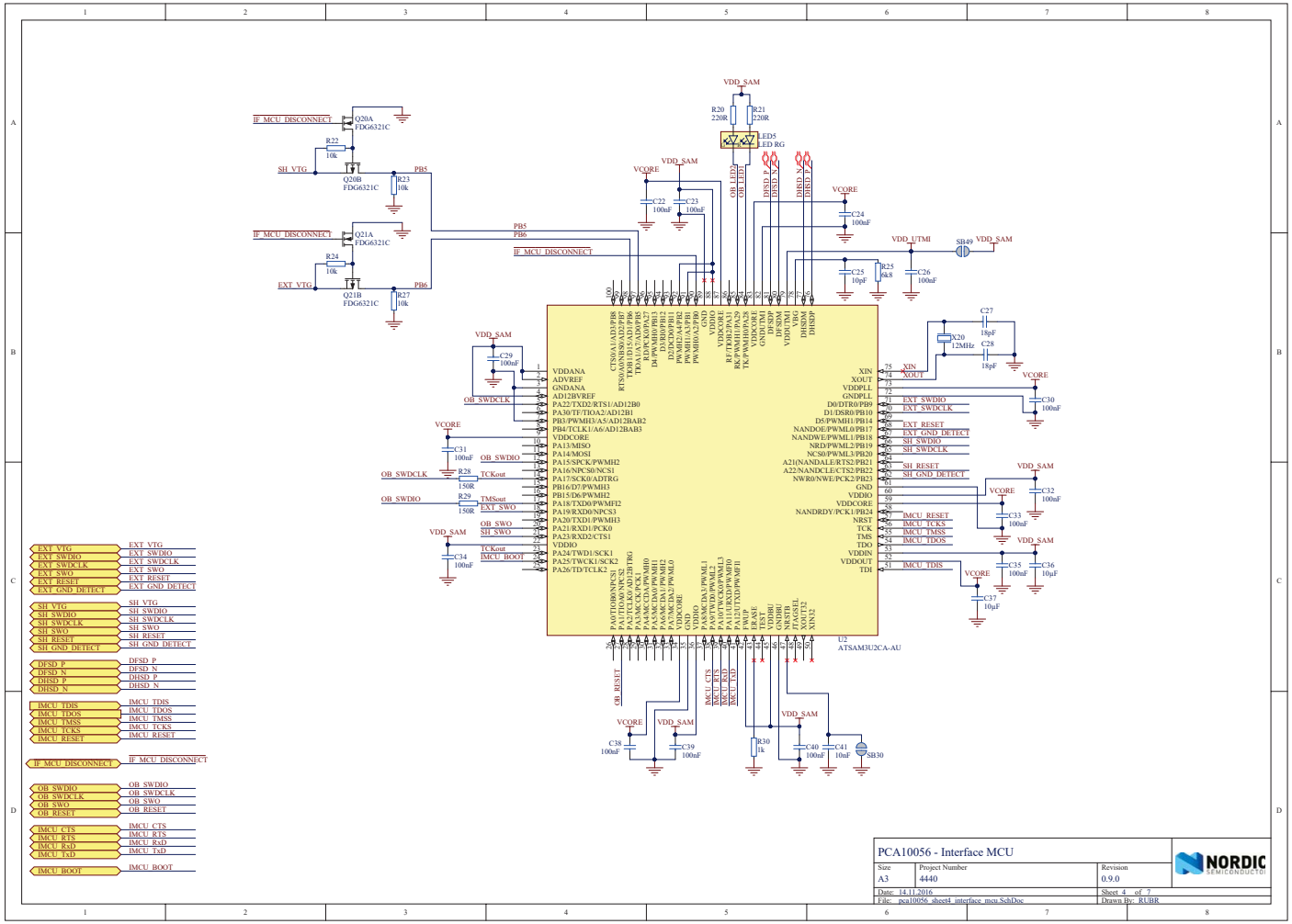
SCH II.1: PCA10056 - Cover, page 1 of 7 [45].



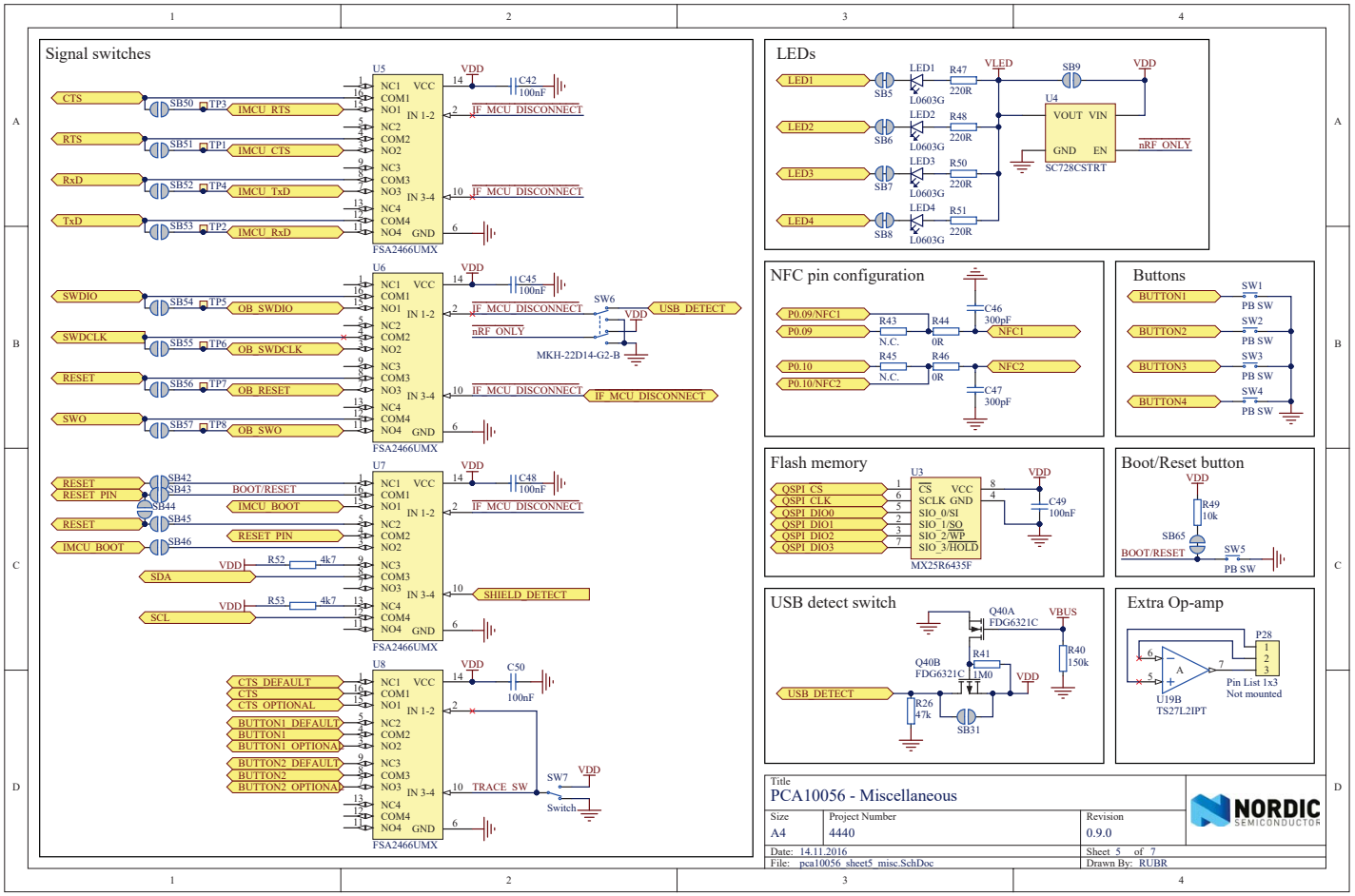
SCH II.2: PCA10056 - nRF52840 SoC, page 2 of 7 [45].



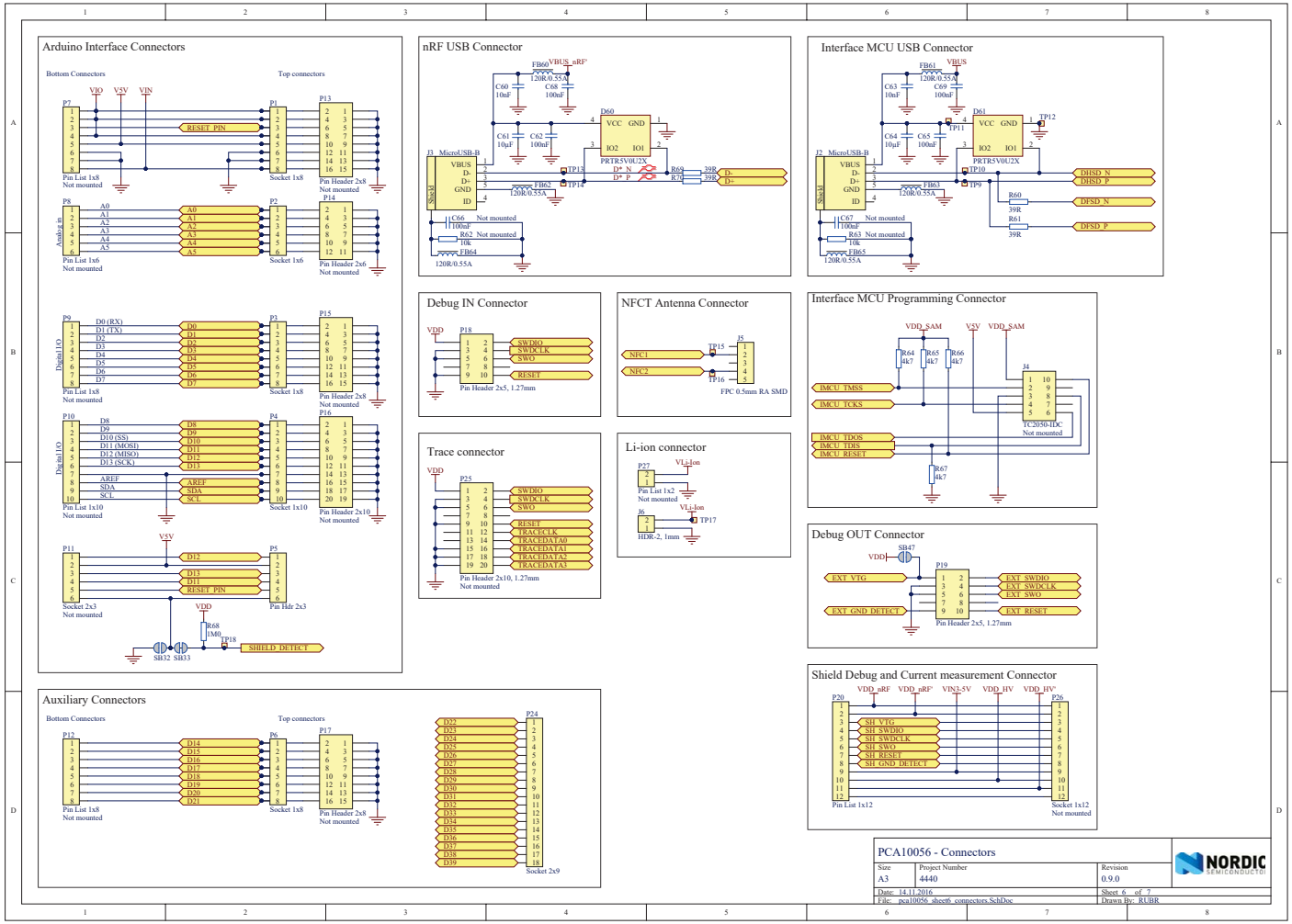
SCH II.3: PCA10056 - Pin map, page 3 of 7 [45].



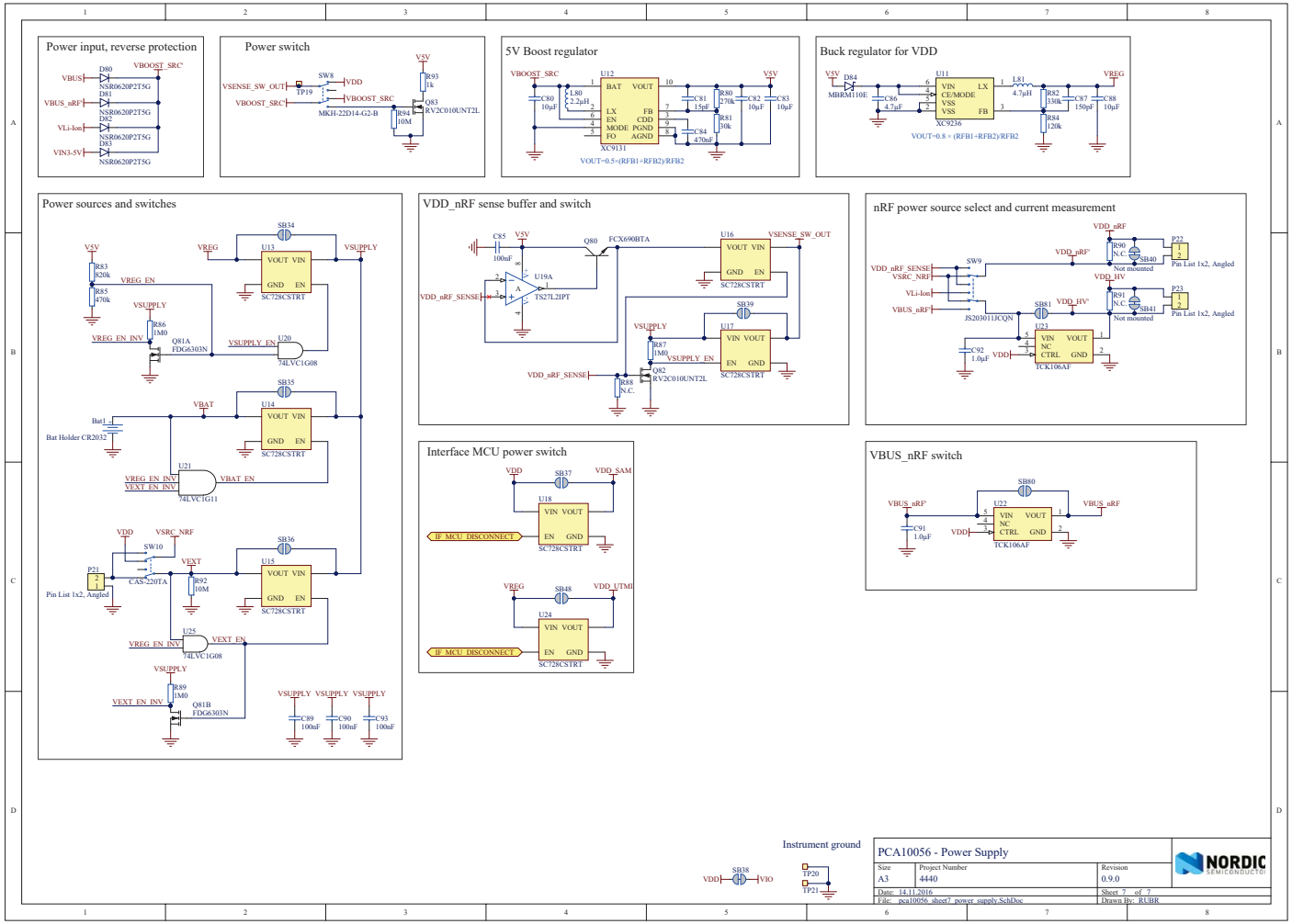
SCH II.4: PCA10056 - Interface MCU, page 4 of 7 [45].



SCH II.5: PCA10056 - Miscellaneous, page 5 of 7 [45].



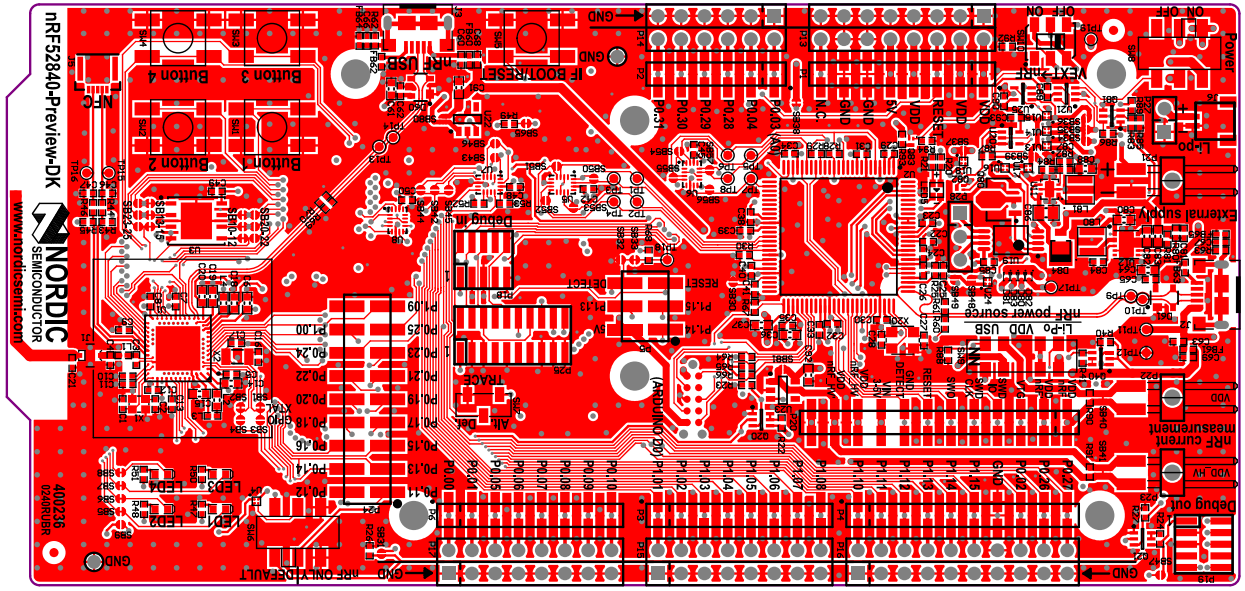
SCH II.6: PCA10056 - Connectors, page 6 of 7 [45].



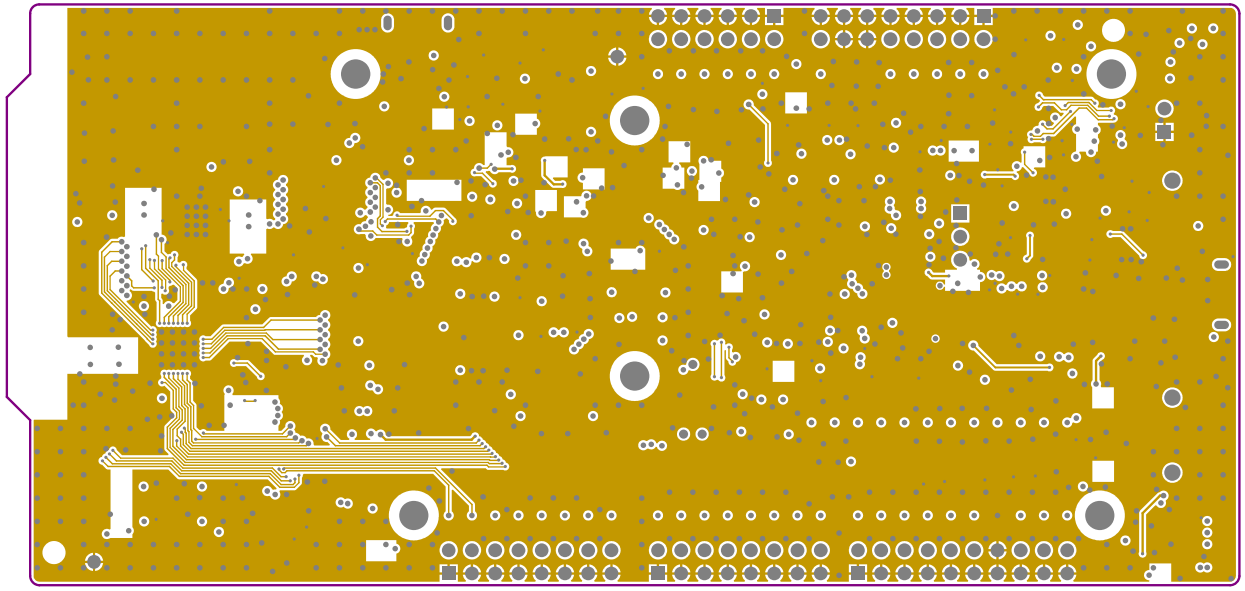
SCH II.7: PCA10056 - Power supply, page 7 of 7 [45].

BOM II.1: nRF52840-QIAA reference design [47].

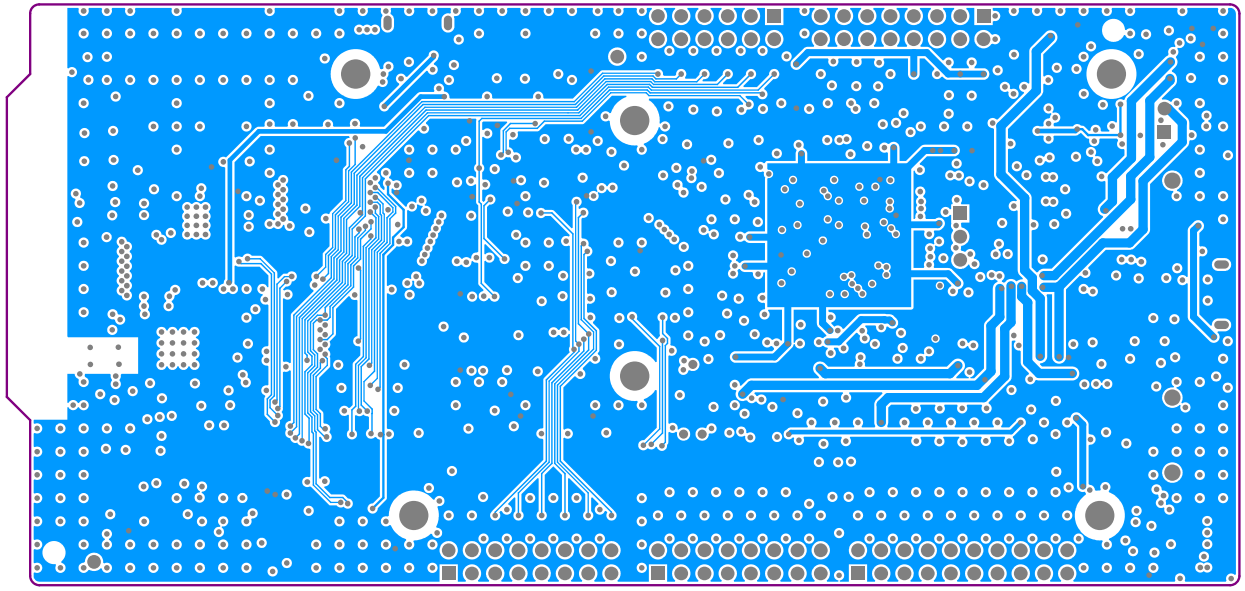
Designator	Value	Description	Footprint
C1, C2, C17, C18	12 pF	Capacitor, NP0, 2 %	0402
C3	0.8 pF	Capacitor, NP0, 10 %	0402
C4	0.5 pF	Capacitor, NP0, 10 %	0402
C5, C7, C8, C12	100 nF	Capacitor, X7R, 10 %	0402
C6, C20	4.7 uF	Capacitor, X7R, 10 %	0603
C9	820 pF	Capacitor, NP0, 5 %	0402
C10, C13, C22	N.C.	Not mounted	0402
C11	100 pF	Capacitor, NP0, 5 %	0402
C14, C15	1.0 uF	Capacitor, X7R, 10 %	0603
C16	47 nF	Capacitor, X7R, 10 %	0402
C19, C21	4.7 uF	Capacitor, X7S, 10 %	0603
L1	4.7 nH	High frequency chip inductor 5 %	0402
L2	10 uH	Chip inductor, IDC, min = 50 mA, 20 %	0603
L3	15 nH	High frequency chip inductor 10 %	0402
L4	10 uH	Chip inductor, IDC, min = 80 mA, 10 %	0603
U1	nRF52840 QIAA	Multi-protocol BLE, IEEE 802.15.4, ANT and 2.4 GHz proprietary SoC	QFN73
X1	32 MHz	XTAL SMD 2016, 32 MHz, Cl=8 pF, Total Tol: 40 ppm	XTAL 2016
X2	32.768 kHz	XTAL SMD 3215, 32.768 kHz, Cl=9 pF, Total Tol: 50 ppm	XTAL 3215



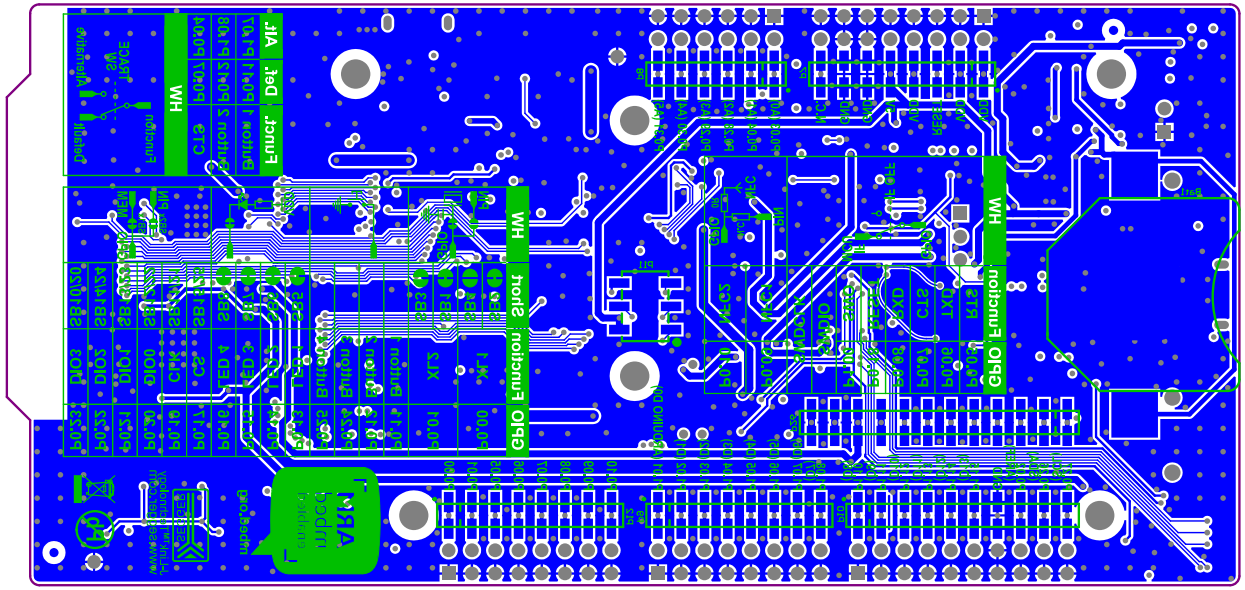
PCB II.1: nRF52840-DK top-layer [45, 59].



PCB II.2: nRF52840-DK mid-layer 1 [45, 59].



PCB II.3: nRF52840-DK mid-layer 2 [45, 59].



PCB II.4: nRF52840-DK bottom-layer [45, 59].

Appendix III: ICMP algorithm and logging C code

```
/* Copyright (c) 2016-2017 Nordic Semiconductor. All Rights Reserved.
 *
 * The information contained herein is property of Nordic Semiconductor ASA.
 * Terms and conditions of usage are described in detail in NORDIC
 * SEMICONDUCTOR STANDARD SOFTWARE LICENSE AGREEMENT.
 *
 * Licensees are granted free, non-transferable use of the information. NO
 * WARRANTY of ANY KIND is provided. This heading must NOT be removed from
 * the file.
 */
 /* ------------------------------------------------------------------------------
 Author         : Roger van Rensburg
 Email          : roger.vanrensburg@wits.ac.za
 Thread version : nRF5_SDK_for_Thread_v0.10.0_e1c3d11
 Description    : Code modified to send ICMP echo requests and log data on an
                  external Arduino data logger shield. This code is intended
                  for source router R1. For source routers R2 to R9, change
                  code addresses on lines 762-821 accordingly. This code is
                  intended for source router R1 and flashed to the nRF52840-DK.
                  For more information on the code files and project repository,
                  please contact the author.
 ------------------------------------------------------------------------------ */
/** @file
 *
 * @defgroup simple coap server example main main.c
 * @{
 * @ingroup simple_coap_server_example_example
 * @brief Simple CoAP Server Example Application main file.
 *
 * @details This example demonstrates a CoAP server application that provides
 resources to control BSP LED 3
 *          via CoAP messages. It can be controlled by a board with related Simple
 CoAP Server application.
 */
#include <assert.h>
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include "bsp_thread.h"
#include "app_timer.h"
#include "nrf_log.h"
#include "nrf_log_ctrl.h"
#include "nrf_delay.h"
#include "nrf_drv_wdt.h"
#include "SEGGER_RTT.h"

/* Modified by author */
#include <openthread/openthread.h>
#include <openthread/diag.h>
#include <openthread/coap.h>
#include <openthread/cli.h>
#include <openthread/thread_ftd.h>
#include <openthread/platform/alarm.h>
#include <openthread/platform/platform.h>

/* Modified by author */
#include "nrf.h"
#include "bsp.h"
#include "ff.h"
#include "diskio_blkdev.h"
#include "nrf_block_dev_sdc.h"

/* Author code from fatfs example START /
// #define NRF_LOG_MODULE_NAME "APP"
#define FILE_NAME               "R1.TXT"
#define STRING1                 "SD card initialized.\r\n"
// SD card SPI pins connection to shield
#define SDC_SCK_PIN             ARDUINO_13_PIN  ///< SDC serial clock (SCK) pin.
#define SDC_MOSI_PIN            ARDUINO_11_PIN  ///< SDC serial data in (DI) pin.
#define SDC_MISO_PIN            ARDUINO_12_PIN  ///< SDC serial data out (DO) pin.
#define SDC_CS_PIN              ARDUINO_10_PIN  ///< SDC chip select (CS) pin.
// #define  NRF_GPIO_PIN_MAP(port, pin)   ((port << 5) | (pin & 0x1F))
```

```
71  #define DATA_WRITE_HEALTHY_PIN           NRF_GPIO_PIN_MAP(1, 5)
72  #define DATA_WRITE_ERROR_PIN             NRF_GPIO_PIN_MAP(1, 4)
73  /* Author code from fatfs example END */
74
75  #define LED_INTERVAL                     100
76  #define PROVISIONING_EXPIRY_TIME         5000
77
78  APP_TIMER_DEF(m_provisioning_timer);
79  APP_TIMER_DEF(m_led_timer);
80  APP_TIMER_DEF(m_led_a_timer_id);
81
82  static FATFS fs;
83  static FIL file;
84  uint32_t bytes_written;
85  FRESULT ff_result;
86
87  /* Author code START */
88
89  char log_data[512];
90  char log_data_all[1024];
91  char log_data1[250];
92  char log_data2[250];
93  char log_data3[250];
94  char log_data4[250];
95  char log_data5[250];
96  char log_data6[250];
97  char log_data7[250];
98  char log_data8[250];
99
100 uint8_t logging_enabled = 0;
101 uint8_t enable_logging = 0;
102 uint8_t cli_count_callback = 0;
103 uint8_t ping_count_callback = 0;
104
105 /* Author code END */
106
107 nrf_drv_wdt_channel_id m_channel_id;
108
109 /**
110  * @brief  SDC block device definition
111  * */
112
113 NRF_BLOCK_DEV_SDC_DEFINE(
114         m_block_dev_sdc,
115         NRF_BLOCK_DEV_SDC_CONFIG(
116                 SDC_SECTOR_SIZE,
117                 APP_SDCARD_CONFIG(SDC_MOSI_PIN, SDC_MISO_PIN, SDC_SCK_PIN, SDC_CS_PIN)
118          ),
119          NFR_BLOCK_DEV_INFO_CONFIG("Nordic", "SDC", "1.00")
120 );
121
122 /*
    ------------------------------------------------------------------------------------ */
123
124 /* Author code from fatfs example START */
125
126 /**
127  * @brief Function for demonstrating FAFTS usage
128  */
129 static void fatfs_init()
130 {
131     static DIR dir;
132     static FILINFO fno;
133     DSTATUS disk_state = STA_NOINIT;
134     // Initialize FATFS disk I/O interface by providing the block device
135     static diskio blkdev t drives[] =
136     {
137             DISKIO_BLOCKDEV_CONFIG(NRF_BLOCKDEV_BASE_ADDR(m_block_dev_sdc,
            block_dev), NULL)
138     };
139     diskio_blockdev_register(drives, ARRAY_SIZE(drives));
140     NRF_LOG_INFO("Initializing disk 0 (SDC)...\r\n");
141     for (uint32_t retries = 3; retries && disk_state; --retries)
```

```
142         {
143             disk_state = disk_initialize(0);
144         }
145         if (disk_state)
146         {
147             NRF_LOG_INFO("Disk initialization failed.\r\n");
148 
149             nrf_gpio_pin_set(DATA_WRITE_ERROR_PIN);
150 
151             return;
152         }
153 
154         uint32_t blocks_per_mb = (1024uL * 1024uL) /
        m_block_dev_sdc.block_dev.p_ops->geometry(&m_block_dev_sdc.block_dev)->blk_size;
155         uint32_t capacity =
        m_block_dev_sdc.block_dev.p_ops->geometry(&m_block_dev_sdc.block_dev)->blk_count
        / blocks_per_mb;
156         NRF_LOG_INFO("Capacity: %d MB\r\n", capacity);
157         NRF_LOG_INFO("Mounting volume...\r\n");
158         ff_result = f_mount(&fs, "", 1);
159 
160         if (ff_result)
161         {
162             NRF_LOG_INFO("Mount failed.\r\n");
163             return;
164         }
165         NRF_LOG_INFO("\r\n Listing directory: /\r\n");
166         ff_result = f_opendir(&dir, "/");
167 
168         if (ff_result)
169         {
170             NRF_LOG_INFO("Directory listing failed!\r\n");
171             return;
172         }
173         do
174         {
175             ff_result = f_readdir(&dir, &fno);
176 
177             if (ff_result != FR_OK)
178             {
179                 NRF_LOG_INFO("Directory read failed.");
180                 return;
181             }
182 
183             if (fno.fname[0])
184             {
185                 if (fno.fattrib & AM_DIR)
186                 {
187                     NRF_LOG_RAW_INFO("   <DIR>   %s\r\n",(uint32_t)fno.fname);
188                 }
189                 else
190                 {
191                     NRF_LOG_RAW_INFO("%9lu  %s\r\n", fno.fsize, (uint32_t)fno.fname);
192                 }
193             }
194         }
195 
196         while (fno.fname[0]);
197         NRF_LOG_RAW_INFO("\r\n");
198     }
199     /*
    ---------------------------------------------------------------------------- */
200     static void fatfs_data()
201     {
202         // NRF_LOG_INFO("Writing to file " FILE_NAME "...");
203         ff_result = f_open(&file, FILE_NAME, FA_READ | FA_WRITE | FA_OPEN_APPEND);
204 
205         if (ff_result != FR_OK)
206         {
207             // NRF_LOG_INFO("Unable to open or create file: " FILE_NAME ".");
208         }
209         ff_result = f_write(&file, log_data_all, sizeof(log_data_all) - 1, (UINT *)
        &bytes_written);
```

```
210
211     if (ff result == FR OK)
212     {
213         // NRF_LOG_INFO("Write failed\r\n.");
214         nrf_gpio_pin_clear(DATA_WRITE_ERROR_PIN);
215
216     }
217     else
218     {
219         // NRF LOG INFO("%d bytes written.", bytes written);
220         nrf gpio pin set(DATA WRITE ERROR PIN);
221     }
222     (void) f_close(&file);
223     // f_sync(&file);
224     return;
225 }
226
227 /* Author code from fatfs example END */
228
229 /*
    ------------------------------------------------------------------------------------------ */
230 static void light_request_handler(void                * p_context,
231                                   otCoapHeader        * p_header,
232                                   otMessage           * p_message,
233                                   const otMessageInfo * p_message_info);
234
235 static void provisioning request handler(void                * p context,
236                                          otCoapHeader        * p_header,
237                                          otMessage           * p_message,
238                                          const otMessageInfo * p_message_info);
239 typedef enum
240 {
241     DEVICE_TYPE_REMOTE_CONTROL,
242     DEVICE TYPE LIGHT
243 } device type t;
244
245 typedef enum
246 {
247     LIGHT_OFF = 0,
248     LIGHT_ON,
249
250     /* Author code START */
251
252     R01_MM_PING_ENABLED = 11,
253     R02_MM_PING_ENABLED = 12,
254     R03_MM_PING_ENABLED = 13,
255     R04_MM_PING_ENABLED = 14,
256     R05_MM_PING_ENABLED = 15,
257     R06_MM_PING_ENABLED = 16,
258     R07 MM PING ENABLED = 17,
259     R08_MM_PING_ENABLED = 18,
260     R09_MM_PING_ENABLED = 19,
261
262     MM_PING_DISABLED = 20,
263
264     /* Author code END */
265
266     LIGHT_TOGGLE
267
268 } light_command_t;
269
270 typedef struct
271 {
272     otInstance     * p_ot_instance;         /**< A pointer to the OpenThread
        instance. */
273     bool             enable provisioning;   /**< Information if provisioning is
        enabled. */
274     uint32_t         provisioning_expiry;   /**< Provisioning timeout time. */
275     otCoapResource   provisioning_resource; /**< CoAP provisioning resource. */
276     otCoapResource   light_resource;        /**< CoAP light resource. */
277 } application_t;
278
279 application_t m_app =
```

```
{
    .p ot instance         = NULL,
    .enable provisioning   = false,
    .provisioning_expiry   = 0,
    .provisioning_resource = {"provisioning", provisioning_request_handler, NULL,
    NULL},
    .light_resource        = {"light", light_request_handler, NULL, NULL},
};

/**************************************************************************************
*
 * @section CoAP

 **************************************************************************************
 /
static void light_on(void)
{
    LEDS ON(BSP LED 3 MASK);
}
static void light_off(void)
{
    LEDS_OFF(BSP_LED_3_MASK);
}
static void light_toggle(void)
{
    LEDS INVERT(BSP LED 3 MASK);
}
static void provisioning_disable(otInstance * p_instance)
{
    m_app.enable_provisioning = false;
    m_app.provisioning_expiry = 0;
    app_timer_stop(m_provisioning_timer);
}
static void provisioning enable(otInstance * p instance)
{
    m_app.enable_provisioning = true;
    m_app.provisioning_expiry = otPlatAlarmGetNow() + PROVISIONING_EXPIRY_TIME;
    app_timer_start(m_provisioning_timer,
                    APP_TIMER_TICKS(PROVISIONING_EXPIRY_TIME),
                    p_instance);
}
static void light response send(void                * p context,
                                otCoapHeader        * p request header,
                                const otMessageInfo * p_message_info)
{
    otError      error = OT_ERROR_NONE;
    otCoapHeader header;
    otMessage  * p_response;

    do
    {
        otCoapHeaderInit(&header, OT_COAP_TYPE_ACKNOWLEDGMENT, OT_COAP_CODE_CHANGED);
        otCoapHeaderSetMessageId(&header, otCoapHeaderGetMessageId(p_request_header));
        otCoapHeaderSetToken(&header,
                             otCoapHeaderGetToken(p_request_header),
                             otCoapHeaderGetTokenLength(p_request_header));

        p response = otCoapNewMessage(p context, &header);
        if (p_response == NULL)
        {
            break;
        }

        error = otCoapSendResponse(p_context, p_response, p_message_info);

    } while (false);

    if (error != OT_ERROR_NONE && p_response != NULL)
    {
        otMessageFree(p_response);
    }
}
static void light_request_handler(void               * p_context,
```

```
                                    otCoapHeader        * p_header,
                                    otMessage           * p_message,
                                    const otMessageInfo * p_message_info)
{
    (void)p_message;
    uint8_t command;
    uint32_t err_code;
    do
    {
        if (otCoapHeaderGetType(p_header) != OT_COAP_TYPE_CONFIRMABLE &&
            otCoapHeaderGetType(p_header) != OT_COAP_TYPE_NON_CONFIRMABLE)
        {
            break;
        }

        if (otCoapHeaderGetCode(p_header) != OT_COAP_CODE_PUT)
        {
            break;
        }

        if (otMessageRead(p_message, otMessageGetOffset(p_message), &command, 1) != 1)
        {
            NRF_LOG_INFO("light handler - missing command\r\n");
        }

        switch (command)
        {
            case LIGHT_ON:
                light_on();
                break;

            case LIGHT_OFF:
                light_off();
                break;

            case LIGHT_TOGGLE:
                light_toggle();
                break;

            /*  Author code START */

            // Select for each router
            case R01_MM_PING_ENABLED:
            /*
               A significant performance degradation in latency measurements and
               packets loss
               are detected when a multicast messages are send from a source router
               to the
               respective destination routers.
               This is perhaps due to MCU buffer overload.
               Therefore, nodes will be randomly activated remotely to start their
               respective
               unicast ICMP requests sequentially.
               *** Correction *** - This was MOSTLY due to the logging.
               Data logging now only occurs after ICMP requests - see main().
            */
                err_code = app_timer_start(m_led_a_timer_id, APP_TIMER_TICKS(700),
                NULL);
                APP_ERROR_CHECK(err_code);

            break;

            // If byte 10 is received from the end-device, stop ICMP ping request
            case MM_PING_DISABLED:
                err_code = app_timer_stop(m_led_a_timer_id);
                APP_ERROR_CHECK(err_code);

                bsp_board_led_off(3);

            break;

            /* Author code END */

```

```
418             default:
419             break;
420         }
421 
422         if (otCoapHeaderGetType(p_header) == OT_COAP_TYPE_CONFIRMABLE)
423         {
424             light_response_send(p_context, p_header, p_message_info);
425         }
426 
427     } while (false);
428 }
429 
430 static otError provisioning_response_send(void                * p_context,
431                                           otCoapHeader        * p_request_header,
432                                           uint8_t               device_type,
433                                           const otMessageInfo * p_message_info)
434 {
435     otError      error = OT_ERROR_NO_BUFS;
436     otCoapHeader header;
437     otMessage  * p_response;
438 
439     do
440     {
441         otCoapHeaderInit(&header, OT_COAP_TYPE_NON_CONFIRMABLE, OT_COAP_CODE_CONTENT);
442         otCoapHeaderSetToken(&header,
443                              otCoapHeaderGetToken(p_request_header),
444                              otCoapHeaderGetTokenLength(p_request_header));
445         otCoapHeaderSetPayloadMarker(&header);
446 
447         p_response = otCoapNewMessage(p_context, &header);
448         if (p_response == NULL)
449         {
450             break;
451         }
452 
453         error = otMessageAppend(p_response, &device_type, 1);
454         if (error != OT_ERROR_NONE)
455         {
456             break;
457         }
458 
459         error = otMessageAppend(p_response, otThreadGetMeshLocalEid(p_context),
        sizeof(otIp6Address));
460         if (error != OT_ERROR_NONE)
461         {
462             break;
463         }
464 
465         error = otCoapSendResponse(p_context, p_response, p_message_info);
466 
467     } while (false);
468 
469     if (error != OT_ERROR_NONE && p_response != NULL)
470     {
471         otMessageFree(p_response);
472     }
473 
474     return error;
475 }
476 
477 static void provisioning_request_handler(void                * p_context,
478                                          otCoapHeader        * p_header,
479                                          otMessage           * p_message,
480                                          const otMessageInfo * p_message_info)
481 {
482     (void)p_message;
483     otMessageInfo message_info;
484 
485     if (!m_app.enable_provisioning)
486     {
487         return;
488     }
489 
```

```
490         if (otCoapHeaderGetType(p_header) == OT_COAP_TYPE_NON_CONFIRMABLE &&
491             otCoapHeaderGetCode(p_header) == OT_COAP_CODE_GET)
492         {
493             message_info = *p_message_info;
494             memset(&message_info.mSockAddr, 0, sizeof(message_info.mSockAddr));
495             if (provisioning_response_send(p_context, p_header, DEVICE_TYPE_LIGHT,
                &message_info) ==
496                 OT_ERROR_NONE)
497             {
498                 provisioning_disable(p_context);
499             }
500         }
501     }
502
503     /**************************************************************************************
    *
504      * @section State change handling
505
     **************************************************************************************
     */
506     static void role_change_handler(void * p_context, otDeviceRole role)
507     {
508         switch(role)
509         {
510             case OT_DEVICE_ROLE_CHILD:
511             case OT_DEVICE_ROLE_ROUTER:
512             case OT_DEVICE_ROLE_LEADER:
513                 break;
514
515             case OT_DEVICE_ROLE_DISABLED:
516             case OT_DEVICE_ROLE_DETACHED:
517             default:
518                 provisioning_disable(p_context);
519                 break;
520         }
521     }
522     static void state_changed_callback(uint32_t flags, void * p_context)
523     {
524         if (flags & OT_CHANGED_THREAD_ROLE)
525         {
526             role_change_handler(p_context, otThreadGetDeviceRole(p_context));
527         }
528
529         NRF_LOG_INFO("State changed! Flags: 0x%08x Current role: %d\r\n", flags,
        otThreadGetDeviceRole(p_context));
530     }
531
532     /**************************************************************************************
    *
533      * @section Buttons
534     **************************************************************************************
    */
535     static void bsp_event_handler(bsp_event_t event)
536     {
537         uint32_t err_code;
538
539         switch (event)
540         {
541             /* Author code START */
542
543             // If button 1 is pressed, start the timer
544             // The timer handler will execute the ICMP requests
545             case BSP_EVENT_KEY_0:
546                 err_code = app_timer_start(m_led_a_timer_id, APP_TIMER_TICKS(700), NULL);
547                 APP_ERROR_CHECK(err_code);
548              break;
549
550              /* Author code *end here */
551
552             case BSP_EVENT_KEY_1:
553             break;
554
555             case BSP_EVENT_KEY_2:
```

```
556                 err_code = app_timer_stop(m_led_a_timer_id);
557                 APP_ERROR_CHECK(err_code);
558                 bsp_board_led_off(3);
559             break;
560 
561         case BSP_EVENT_KEY_3:
562             provisioning_enable(m_app.p_ot_instance);
563         break;
564 
565         default:
566             return;
567     }
568 }
569 /**************************************************************************************************
    *
570  * @section Timers
571 
     **************************************************************************************************
     /
572 static void provisioning_timer_handler(void * p_context)
573 {
574     provisioning_disable(p_context);
575 }
576 
577 static void led_timer_handler(void * p_context)
578 {
579     (void)p_context;
580     if (m_app.enable_provisioning)
581     {
582         LEDS_INVERT(BSP_LED_2_MASK);
583     }
584     else
585     {
586         LEDS_OFF(BSP_LED_2_MASK);
587     }
588 }
589 
590 /* Author code START */
591 
592 int console_output(const char *p_buf, uint16_t buf_length, void *p_context)
593 {
594     uint32_t err_code;
595     (void)p_context;
596     // We need to ensure that p_buf is null terminated string in order to use
    NRF_LOG_INFO
597     // macro to print it.
598     // Currently OT always return p_buf as null-terminated string so below lines could
599     // be removed. But there is possibility that this will change over time.
600     char str[512];
601     char str1[30];
602     char str2[30];
603     char str3[30];
604     char str4[30];
605     char str5[30];
606 
607     memcpy(str, p_buf, buf_length);
608     str[buf_length] = 0;
609 
610     // If logging is not enabled, display other CLI commands in RTT viewer
611     if (logging_enabled == 0) NRF_LOG_INFO("%s", (uint32_t)str);
612 
613     // An ICMP request (PING) from a single node, calls the console_output handler 5
    times
614     // For two nodes, calls the console_output handler 10 times
615     if (logging_enabled == 1)
616     {
617         cli_count_callback++;
618 
619         if (cli_count_callback == 1)
620         {
621             // 1st string of ping receive from a router
622             strcpy(str1, str);
623             // NRF_LOG_INFO("1: %s %d", (uint32_t)str1, cli_count_callback);
```

```
624         }
625
626         if (cli count callback == 2)
627         {
628             // 2nd string of ping receive from a router
629             strcpy(str2, str);
630             // NRF_LOG_INFO("2:%s", (uint32_t)str2);
631         }
632
633         if (cli count callback == 3)
634         {
635             // 3rd string of ping receive from a router
636             strcpy(str3, str);
637             // NRF_LOG_INFO("3:%s", (uint32_t)str3);
638         }
639
640         if (cli_count_callback == 4)
641         {
642             // 4th string of ping receive from a router
643             strcpy(str4, str);
644             // NRF_LOG_INFO("4:%s", (uint32_t)str4);
645         }
646
647         if (cli_count_callback == 5)
648         {
649             // 5th string of ping receive from a router
650             strcpy(str5, str);
651             // NRF_LOG_INFO("5:%s", (uint32_t)str5);
652
653             // Reset cli_count_callback
654             cli_count_callback = 0;
655             // Concatenate strings 1-5
656             sprintf(log_data, "%s %s %s %s %s\r\n", str1, str2, str3, str4, str5);
657
658             ping count callback++;
659             // NRF_LOG_INFO("ping_count_callback: %d ", ping_count_callback);
660
661             if (ping_count_callback == 1)
662             {
663                 strcpy(log_data1, log_data);
664                 NRF_LOG_INFO("1: %s", (uint32_t)log_data1);
665             }
666             if (ping count callback == 2)
667             {
668                 strcpy(log_data2, log_data);
669                 NRF_LOG_INFO("2: %s", (uint32_t)log_data2);
670             }
671             if (ping_count_callback == 3)
672             {
673                 strcpy(log data3, log data);
674                 NRF_LOG_INFO("3: %s", (uint32_t)log_data3);
675             }
676             if (ping_count_callback == 4)
677             {
678                 strcpy(log_data4, log_data);
679                 NRF_LOG_INFO("4: %s", (uint32_t)log_data4);
680             }
681             if (ping count callback == 5)
682             {
683                 strcpy(log_data5, log_data);
684                 NRF_LOG_INFO("5: %s", (uint32_t)log_data5);
685             }
686             if (ping_count_callback == 6)
687             {
688                 strcpy(log data6, log data);
689                 NRF LOG INFO("6: %s", (uint32 t)log data6);
690             }
691             if (ping_count_callback == 7)
692             {
693                 strcpy(log_data7, log_data);
694                 NRF_LOG_INFO("7: %s", (uint32_t)log_data7);
695             }
696             if (ping_count_callback == 8)
```

```
            {
                strcpy(log data8, log data);
                NRF LOG INFO("8: %s", (uint32 t)log data8);

                // Stop the timer which requests ICMP requests for data logging
                err_code = app_timer_stop(m_led_a_timer_id);
                APP_ERROR_CHECK(err_code);

                ping_count_callback = 0;

                enable logging = 1;
                nrf_gpio_pin_set(DATA_WRITE_HEALTHY_PIN);
            }
        }
    }
    return buf_length;
}

/* Author code END */

static void timer_handler(void * p_context)
{
    // Author code START

    // char REMOTE_EID_ADDRESS[100] = "fdde:ad00:beef:0:51ac:188c:2785:bcf2";
    // char R1 EID ADDRESS    [100] = "ping fdde:ad00:beef:0:8030:6e5b:f43e:1000";
    char R2 EID ADDRESS    [100] = "ping fdde:ad00:beef:0:bd0:bee:c8e2:c840 1024";
    char R3_EID_ADDRESS    [100] = "ping fdde:ad00:beef:0:bc44:dcd2:d1ec:d43e 1024";
    char R4_EID_ADDRESS    [100] = "ping fdde:ad00:beef:0:5ed8:e4e9:1eb2:f71 1024";
    char R5_EID_ADDRESS    [100] = "ping fdde:ad00:beef:0:4b3b:c4c6:f88f:57b1 1024";
    char R6_EID_ADDRESS    [100] = "ping fdde:ad00:beef:0:9d06:e7f8:1ab0:b8a8 1024";
    char R7_EID_ADDRESS    [100] = "ping fdde:ad00:beef:0:94c9:44b1:9ce8:f5ae 1024";
    char R8_EID_ADDRESS    [100] = "ping fdde:ad00:beef:0:bcdd:cd99:f4e2:59cf 1024";
    char R9 EID ADDRESS    [100] = "ping fdde:ad00:beef:0:86e:801e:fe28:6d8 1024";

    static int timer_handler_count = 0;
    timer_handler_count++;
    LEDS_INVERT(BSP_LED_3_MASK);

    if (timer_handler_count == 1)
    {
        otCliConsoleInit(m app.p ot instance, console output, NULL);
        otCliConsoleInputLine(R2 EID ADDRESS, strlen(R2 EID ADDRESS));
    }

    if (timer_handler_count == 2)
    {
        otCliConsoleInit(m_app.p_ot_instance, console_output, NULL);
        otCliConsoleInputLine(R3_EID_ADDRESS, strlen(R3_EID_ADDRESS));
    }

    if (timer_handler_count == 3)
    {
        otCliConsoleInit(m_app.p_ot_instance, console_output, NULL);
        otCliConsoleInputLine(R4_EID_ADDRESS, strlen(R4_EID_ADDRESS));
    }

    if (timer handler count == 4)
    {
        otCliConsoleInit(m_app.p_ot_instance, console_output, NULL);
        otCliConsoleInputLine(R5_EID_ADDRESS, strlen(R5_EID_ADDRESS));
    }

    if (timer_handler_count == 5)
    {
        otCliConsoleInit(m app.p ot instance, console output, NULL);
        otCliConsoleInputLine(R6_EID_ADDRESS, strlen(R6_EID_ADDRESS));
    }

    if (timer_handler_count == 6)
    {
        otCliConsoleInit(m_app.p_ot_instance, console_output, NULL);
        otCliConsoleInputLine(R7_EID_ADDRESS, strlen(R7_EID_ADDRESS));
```

```
770         }
771
772         if (timer handler count == 7)
773         {
774             otCliConsoleInit(m_app.p_ot_instance, console_output, NULL);
775             otCliConsoleInputLine(R8_EID_ADDRESS, strlen(R8_EID_ADDRESS));
776         }
777
778         if (timer_handler_count == 8)
779         {
780             otCliConsoleInit(m app.p ot instance, console output, NULL);
781             otCliConsoleInputLine(R9_EID_ADDRESS, strlen(R9_EID_ADDRESS));
782
783             timer_handler_count = 0;
784         }
785
786         NRF_LOG_INFO("******************************\n");
787
788         /* Author code END */
789     }
790
791     /**************************************************************************************
    *
792     @section Initialization
793     **************************************************************************************
    /
794     static void thread init(void)
795     {
796         otInstance * p_instance;
797
798         PlatformInit(0, NULL);
799
800         p_instance = otInstanceInit();
801         assert(p instance);
802
803         /* Modified by author */
804
805         // otCliUartInit(p_instance);
806         // Enable diagnostic features
807         otDiagInit(p_instance);
808
809         NRF LOG INFO("Thread version: %s\r\n", (uint32 t)otGetVersionString());
810         NRF LOG INFO("Network name:   %s\r\n",
        (uint32_t)otThreadGetNetworkName(p_instance));
811
812         assert(otSetStateChangedCallback(p_instance, &state_changed_callback,
        p_instance) == OT_ERROR_NONE);
813
814         if (!otDatasetIsCommissioned(p_instance))
815         {
816             assert(otLinkSetChannel(p_instance, THREAD_CHANNEL) == OT_ERROR_NONE);
817             assert(otLinkSetPanId(p_instance, THREAD_PANID) == OT_ERROR_NONE);
818         }
819
820         assert(otIp6SetEnabled(p_instance, true) == OT_ERROR_NONE);
821         assert(otThreadSetEnabled(p_instance, true) == OT_ERROR_NONE);
822
823         m app.p ot instance = p instance;
824     }
825     static void coap_init()
826     {
827         m_app.light_resource.mContext = m_app.p_ot_instance;
828         m_app.provisioning_resource.mContext = m_app.p_ot_instance;
829
830         assert(otCoapStart(m app.p ot instance, OT DEFAULT COAP PORT) == OT ERROR NONE);
831         assert(otCoapAddResource(m app.p ot instance, &m app.light resource) ==
        OT_ERROR_NONE);
832         assert(otCoapAddResource(m_app.p_ot_instance, &m_app.provisioning_resource) ==
        OT_ERROR_NONE);
833     }
834     static void timer_init(void)
835     {
836         uint32_t err_code = app_timer_init();
```

```
837       APP_ERROR_CHECK(err_code);
838
839       app_timer_create(&m_provisioning_timer, APP_TIMER_MODE_SINGLE_SHOT,
          provisioning_timer_handler);
840       app_timer_create(&m_led_timer, APP_TIMER_MODE_REPEATED, led_timer_handler);
841
842       app_timer_create(&m_led_a_timer_id, APP_TIMER_MODE_REPEATED, timer_handler);
843
844   }
845   static void thread_bsp_init(void)
846   {
847       uint32_t err_code = bsp_init(BSP_INIT_LED | BSP_INIT_BUTTONS, bsp_event_handler);
848       APP_ERROR_CHECK(err_code);
849
850       err_code = bsp_thread_init(m_app.p_ot_instance);
851       APP_ERROR_CHECK(err_code);
852   }
853
854   static void leds_init(void)
855   {
856       LEDS_CONFIGURE(LEDS_MASK);
857       LEDS_OFF(LEDS_MASK);
858
859       app_timer_start(m_led_timer, APP_TIMER_TICKS(LED_INTERVAL), NULL);
860   }
861   /**
862    * @brief WDT events handler.
863    */
864   void wdt_event_handler(void)
865   {
866       bsp_board_leds_off();
867       // NOTE: The max amount of time we can spend in WDT interrupt is two cycles of
          32768[Hz] clock - after that, reset occurs
868   }
869   /**************************************************************************************
      *
870    * @section Main
871
       **************************************************************************************
       /
872   int main(int argc, char *argv[])
873   {
874       uint32_t err_code;
875       NRF_LOG_INIT(NULL);
876       thread_init();
877       coap_init();
878       timer_init();
879       thread_bsp_init();
880       leds_init();
881       bsp_board_leds_init();
882       APP_ERROR_CHECK(NRF_LOG_INIT(NULL));
883       NRF_LOG_INFO("\r\nFATFS.\r\n\r\n");
884
885       /* Author code START */
886
887       // Configure digital output pins for switching of L1 and L2 on SD card shield
888       // Configure P1.4 as digitla output with no pull-up resistor
889       nrf_gpio_cfg(DATA_WRITE_HEALTHY_PIN,
890                    NRF_GPIO_PIN_DIR_OUTPUT,
891                    NRF_GPIO_PIN_INPUT_DISCONNECT,
892                    NRF_GPIO_PIN_NOPULL,
893                    NRF_GPIO_PIN_H0H1 ,
894                    NRF_GPIO_PIN_NOSENSE);
895       // Configure P1.5 as digital output with no pull-up resistor
896       nrf_gpio_cfg(DATA_WRITE_ERROR_PIN,
897                    NRF_GPIO_PIN_DIR_OUTPUT,
898                    NRF_GPIO_PIN_INPUT_DISCONNECT,
899                    NRF_GPIO_PIN_NOPULL,
900                    NRF_GPIO_PIN_H0H1 ,
901                    NRF_GPIO_PIN_NOSENSE);
902       // Clear P1.4 and P1.5 upon MCU start-up - led display on SD shield
903       nrf_gpio_pin_clear(DATA_WRITE_HEALTHY_PIN);
904       nrf_gpio_pin_clear(DATA_WRITE_ERROR_PIN);
```

```
905     // Initialise SD - Status displayed in Segger RT Viewer
906     fatfs init();
907     nrf delay ms(200);
908     // This function sets whether or not the Router Role is enabled
909     otThreadSetRouterRoleEnabled(m_app.p_ot_instance, true);
910     // This function sets the maximum transmit power in dBm
911     otLinkSetMaxTransmitPower(m_app.p_ot_instance, 8);
912     // Configure WDT
913     nrf_drv_wdt_config_t config = NRF_DRV_WDT_DEAFULT_CONFIG;
914     err code = nrf drv wdt init(&config, wdt event handler);
915     APP ERROR CHECK(err code);
916     err_code = nrf_drv_wdt_channel_alloc(&m_channel_id);
917     APP_ERROR_CHECK(err_code);
918     nrf_drv_wdt_enable();
919     // Always enable data looging
920     logging_enabled = 1;
921
922     /* Author code END  */
923
924     char c = '0';
925     err_code = app_timer_start(m_led_a_timer_id, APP_TIMER_TICKS(700), NULL);
926
927     while (true)
928     {
929         otTaskletsProcess(m_app.p_ot_instance);
930         PlatformProcessDrivers(m app.p ot instance);
931
932         /* Author code START */
933
934         if (enable_logging)
935         {
936             sprintf(log_data_all, "%s \n %s \n %s \n %s \n %s \n %s \n %s \n %s \n
                ", log_data1, log_data2, log_data3, log_data4, log_data5, log_data6,
                log data7, log data8);
937             fatfs data();
938             err_code = app_timer_start(m_led_a_timer_id, APP_TIMER_TICKS(700), NULL);
939             APP_ERROR_CHECK(err_code);
940             enable_logging = 0;
941             nrf_gpio_pin_clear(DATA_WRITE_HEALTHY_PIN);
942         }
943         c = SEGGER_RTT_GetKey(); // Block until data is available
944             // Manual test to ping all nodes in the network by pressing 'a' on
                keyboard using Segger RT Viewer
945
946             if (c == 'a')
947             {
948                 // SEGGER_RTT_WriteString(0, "Get key character 'a' pressed\r\n\n");
949                 otCliConsoleInit(m_app.p_ot_instance, console_output, NULL);
950
951                 // ping FF03::1 - command for ICMP requests to all mesh local
                    addresses in the network
952                 char command[] = "ping FF03::1\n";
953                 otCliConsoleInputLine(command, strlen(command));
954             }
955
956             // Display node IPv6 address
957             if (c == 'c')
958             {
959                 // Disable logging
960                 logging_enabled = 0;
961                 otCliConsoleInit(m_app.p_ot_instance, console_output, NULL);
962                 char command[] = "ipaddr";
963                 otCliConsoleInputLine(command, strlen(command));
964                 // Enable logging
965                 logging enabled = 1;
966             }
967
968             // Display node extended address
969             if (c == 'd')
970             {
971                 // Disable logging
972                 logging_enabled = 0;
973                 otCliConsoleInit(m_app.p_ot_instance, console_output, NULL);
```

```
974                     char command[] = "extaddr";
975                     otCliConsoleInputLine(command, strlen(command));
976                     // Enable logging
977                     logging_enabled = 1;
978                 }
979 
980                 // Simulate a watchdog reset when pressing a key from PC
981                 if (c == 'e')
982                 {
983                     // Go in an infinite loop
984                     NRF_LOG_INFO("\rInfinite loop triggered for keyboard...\r\n\r\n");
985                     while(true)
986                     {
987                         // Infinite loop...
988                     };
989                 }
990 
991             NRF_LOG_FLUSH();
992             // Feed the dog
993             nrf_drv_wdt_channel_feed(m_channel_id);
994 
995             /* Author code END */
996         }
997     }
998     /**
999      *@}
1000     **/
```

Appendix IV:

Gaussian kernel Matlab script

Author: Roger van Rensburg, Date created: 2018-08-08, Description: Implimentation of Equation 5.1

Contents

- Gaussian kernel distribution route R1 -> R2


```
clc
clear
close
```

Gaussian kernel distribution route R1 -> R2

```
load R1_R2_normal_data_mat

latency = R1_R2_normal_data_mat(:,1);

R1_R2_latency_minmax = minmax(latency');

p = linspace(0,R1_R2_latency_minmax(2),1000);

% Bandwidth
h = 0.8;

% Sample size
n = size(p,2);

% Kernel density estimation
c = 1/sqrt(2*pi*h^2);
f = zeros(size(p,1));

for i=1:n
        f = f + 1/n*c*exp(-(latency(i)-p).^2/(2*h^2));
end

plot(p,f)
```