

**Investigation and development of a system for  
secure synchronisation of information in a  
wireless mesh network**

**Daniel Nicholas de Bruyn**

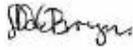
**Dissertation for the degree of  
Magister Technologiae Engineering: Electrical  
in the  
School of Electrical and Computer Systems  
Engineering  
of the  
Faculty of Engineering and Information Technology  
at the  
Central University of Technology, Free State  
December 2010**

**Study leader: Mr BJ Kotze, M Tech. (Eng)**

**Co-study leader: Prof HJ Vermaak, Ph.D.**

## Declaration of independent work

I, Daniel Nicolas de Bruyn, hereby declare that this research project submitted for the degree  
MAGISTER TECHNOLOGIAE: ENGINEERING: ELECTRICAL, is my own independent work that  
has not been submitted before to any institution by anyone or me else as part of any qualification.



.....

SIGNATURE OF STUDENT

18 January 2011.....

DATE

## **Acknowledgments**

I firstly want to thank our Heavenly Father for giving me the ability to complete this study. Secondly, I want to thank my wife and children for their tremendous support and understanding during this study. I would like to thank the following individuals and institutions in particular:

- Special thanks to Ben Kotze for his supervision, guidance and advice.
- Herman Vermaak for his supervision, guidance and advice.
- The Central University of Technology for the financial support.
- My Father for providing me with opportunities in life and for always believing in me.
- Henry Rehne for proof reading the document.

## **Abstract**

This dissertation gives an overview of the research done in developing a protocol to synchronise information in a secure wireless mesh network. Alternative methods to control wireless devices were investigated in the non-controlled frequency spectrum.

The aim of the research was to develop a protocol that can be loaded on a micro-controller with limited intelligence, controlling endpoints. The protocol minimises human interference and automatically negotiates which device becomes the master controller. The device is able to discover and locate neighbour devices in range. The device has the capability to be stationary or mobile and host multiple control endpoints.

Control endpoints can be digital or analogue, input or output, and belongs to a group like security, lighting or irrigation. These capabilities can change according to the solution's requirements. Control endpoints with the same capabilities must be able to establish a connection between each other. An endpoint has a user-friendly name and can update the remote endpoint with the description. When a connection is established both endpoints update each other with their user-friendly name and their status. A local endpoint can trigger a certain action on a receiving control point.

The system was tested with a building monitoring system because it is static and a less expensive choice, thus making the evaluation more suitable. A simulator for a personal computer was developed to evaluate the new protocol. Finally, the protocol was implemented and tested on a micro-controller platform.

## Uittreksel

Die verhandeling gee 'n oorsig van die navorsing wat gedoen is in die ontwikkeling van 'n program wat inligting kan sinkroniseer in 'n radiobeveiligte “mesh”-netwerk. Alternatiewe metodes is om radiobeheerde eenhede te beheer in 'n nie-gekontroleerde radiospektrum.

Die doel van die navorsing is om 'n protokol vir 'n mikro-proseseerder, met beperkte intelligensie as beheeroplossing te ontwikkel. Die protokol beperk menslike opstelling tot 'n minimum en onderhandel outomaties om te bepaal watter eenheid die hoofbeheerder is. Die eenhede kan nabuurige eenhede binne 'n sekere afstand opspoor en kan staties of beweegbaar wees. Die protokol kan veelvuldige beheerpunte per eenheid beheer.

Die beheerpunte kan digitaal of analoog wees, inset of uitset en moet aan 'n groep behoort. Groepe kan sekuriteit, beligting of besproeiing insluit. Hierdie funksies van die beheerpunte kan verander word na gelang van dit wat van die eenheid vereis word. Soortgelyke beheerpunte met dieselfde funksies moet in staat wees om met mekaar in verbinding te kan tree. Elke beheerpunt het 'n gebruikersvriendelike beskrywing wat na die ander beheerpunt opgedateer word. Wanneer 'n verbinding bewerkstellig word sal beide beheerpunte met hul status en gebruikersvriendelike beskrywing opgedateer word. 'n Plaaslike beheerpunt kan die status van die ander beheerpunt verander.

Die stelsel is met 'n boubeheerstelsel getoets aangesien dit staties is en sodoende die evaluasie daarvan vergemaklik. 'n Rekenaarsimulasieprogram is ontwikkel om die nuwe protokol te evalueer. Laastens is die protokol op 'n mikrobeheerde platform geïmplementeer en getoets.

# Table of contents

Declaration of independent work.....	ii
Acknowledgments .....	iii
Abstract.....	iv
Uittreksel.....	v
List of figures.....	xi
List of tables.....	xvii
Acronyms and abbreviations.....	xix
1 Introduction.....	1
1.1 Hypothetical solution.....	2
1.2 Expected outcomes .....	2
1.3 Layout of dissertation.....	4
2 Communication principles.....	6
2.1 Open Systems Interconnection Reference Model.....	6
2.1.1 Physical layer.....	7
2.1.2 Data Link layer.....	7
2.1.3 Network layer.....	8
2.1.4 Transport layer.....	8
2.1.5 Session layer .....	8
2.1.6 Presentation.....	8
2.1.7 Application.....	9
2.2 Network communication fundamentals .....	9
2.2.1 Network topologies .....	9

2.2.2	Collisions .....	12
2.2.3	Communication methods .....	13
2.3	Integrity of data.....	14
2.4	Protocols for Wireless Personal Area Networks .....	17
2.4.1	IEEE 802.15.4 .....	17
2.4.2	MiWi™ Wireless Network Protocol Stack .....	22
2.4.3	Zigbee .....	26
2.5	Summary .....	33
3	Standard development of a secure synchronisation of an information system in a wireless mesh network .....	34
3.1	Basic understanding .....	34
3.1.1	Requirements .....	34
3.1.2	Discover manageable logical topology from a Mesh topology.....	34
3.1.3	Send packets to correct destination .....	39
3.1.4	Devices control each other.....	40
3.2	Commands .....	40
3.2.1	Overview .....	40
3.2.2	Device discovery and maintenance.....	42
3.2.3	Establishing connections and maintenance.....	52
3.3	Summary .....	61
4	Device discovery and maintenance.....	63
4.1	Simulate wireless .....	63
4.2	Layout of program .....	63
4.2.1	Endpoint setup .....	64

4.2.2	Global setup .....	65
4.2.3	Trace .....	66
4.2.4	Simulate .....	67
4.3	Cycle redundancy check .....	67
4.4	Control frames .....	68
4.5	Discover short address function development .....	69
4.5.1	Frame format.....	69
4.5.2	Algorithms .....	71
4.6	Keepalive function development .....	82
4.6.1	Frame format.....	82
4.6.2	Algorithms .....	85
4.7	Simulation and results.....	89
4.7.1	Three devices request short addresses in same broadcast domain .....	89
4.7.2	Cleanup parent-child table .....	91
4.7.3	New coordinator selected.....	92
4.7.4	Two coordinators detected .....	94
4.7.5	New coordinator forced .....	96
4.8	Conclusion .....	97
5	Maintaining connections between control points .....	98
5.1	Connection frames .....	98
5.2	Forward function development.....	100
5.2.1	Algorithms .....	101
5.3	Resolve Medium Access Control address function development.....	103
5.3.1	Frame format.....	104

5.3.2	Algorithms .....	105
5.4	Management of connections function development .....	110
5.4.1	Connection tables.....	110
5.4.2	Frame format.....	112
5.4.3	Algorithms .....	116
5.5	Management of endpoint status function development .....	126
5.5.1	Frame format.....	126
5.5.2	Algorithms .....	128
5.6	Simulation and results.....	134
5.6.1	Unsuccessful connection established between two endpoints .....	136
5.6.2	Successful connection established between two endpoints.....	137
5.6.3	Devices changes short addresses.....	140
5.6.4	One output endpoint linked to two input endpoints .....	145
5.6.5	Analogue endpoints.....	151
5.6.6	Delete connection between two endpoints.....	153
5.7	Conclusion .....	154
6	Constructing of the device controller.....	156
6.1	Block diagram.....	156
6.2	Choosing the correct components .....	157
6.3	Developing the PIC18LF462 microcontroller software.....	159
6.3.1	Interrupt handling.....	159
6.3.2	Configuring the USART .....	160
6.3.3	Redesign the Request connection process.....	164
6.3.4	Redesign the Manage connection process.....	169

6.3.5	Contact de-bouncing .....	171
6.3.6	Device circuit diagram .....	172
6.4	RFD21733 Wireless Transceiver .....	174
6.4.1	Wireless circuit diagram .....	174
6.4.2	Device and modem hardware .....	175
6.4.3	Collision avoidance .....	177
6.5	Final test results .....	182
6.6	Security of data .....	195
6.7	Conclusion .....	195
7	Conclusion .....	196
	References .....	198
	Appendix A: MRF2470 Features .....	201

## List of figures

Figure 1.1	Typical home environment connected as a network.....	3
Figure 2.1	Seven-Layer OSI model.....	6
Figure 2.2	OSI relationship.....	7
Figure 2.3	Bus topology.....	9
Figure 2.4	Ring topology.....	10
Figure 2.5	Star topology.....	10
Figure 2.6	Extended Star.....	11
Figure 2.7	Hierarchical topology.....	11
Figure 2.8	Fully Mesh topology.....	12
Figure 2.9	Partial Mesh topology.....	12
Figure 2.10	AES ShiftRow process.....	17
Figure 2.11	AES MixColumn process.....	17
Figure 2.12	IEEE 802.15.4 Protocol stack.....	18
Figure 2.13	IEEE 802.15.4 Star topology.....	19
Figure 2.14	IEEE 802.15.4 Peer-to-Peer topology.....	20
Figure 2.15	IEEE 802.15.4 Multi Cluster Network.....	21
Figure 2.16	MiWi Star topology.....	23
Figure 2.17	MiWi Cluster Tree topology.....	23
Figure 2.18	MiWi Mesh topology.....	24
Figure 2.19	MiWi short address allocation.....	25
Figure 2.20	Zigbee Frequency Bands.....	28
Figure 2.21	Zigbee Star topology.....	29
Figure 2.22	Zigbee Tree topology.....	30
Figure 2.23	Zigbee Mesh topology.....	30

Figure 2.24	Zigbee OSI model.....	31
Figure 3.1	Wireless topology shown as antenna connectivity with perfect sphere.....	35
Figure 3.2	Wireless topology shown with lines.....	36
Figure 3.3	Short address structure.....	37
Figure 3.4	Same network with different coordinators.....	38
Figure 3.5	Logical Tree topology routing.....	40
Figure 3.6	Overview of connections.....	42
Figure 3.7	Request short address with two devices.....	43
Figure 3.8	Request short address with three devices.....	44
Figure 3.9	One-device join two Personal Area Networks together.....	45
Figure 3.10	Coordinator pre-defined.....	47
Figure 3.11	Timing schedule for Request short address procedure.....	48
Figure 3.12	Keepalive illustration.....	50
Figure 3.13	Request short address and Keepalive commands builds topology example 1.....	51
Figure 3.14	Request short address and Keepalive commands builds topology example 2.....	51
Figure 3.15	Routing decisions.....	52
Figure 3.16	Light switch.....	54
Figure 3.17	Passage light switch.....	55
Figure 3.18	Irrigation system with rain sensor.....	55
Figure 3.19	Establish connection successfully.....	56
Figure 3.20	Terminate connection.....	57
Figure 3.21	Send status.....	59
Figure 3.22	MAC Broadcast rediscovers changed short addresses.....	60
Figure 3.23	Broadcast storm.....	61
Figure 3.24	Broadcast avoidance.....	61
Figure 4.1	Define the endpoints.....	64

Figure 4.2	Global setup .....	65
Figure 4.3	Trace .....	66
Figure 4.4	Simulate .....	67
Figure 4.5	XOR Gate .....	68
Figure 4.6	CRC-16 generator .....	68
Figure 4.7	Control frame .....	68
Figure 4.8	Discover short address frame .....	69
Figure 4.9	Request short address frame sample .....	70
Figure 4.10	Offer short address frame sample .....	70
Figure 4.11	Acknowledge short address frame sample .....	70
Figure 4.12	Reset short address sample .....	70
Figure 4.13	Protect coordinator short address frame sample .....	71
Figure 4.14	Flow diagram of the main program .....	72
Figure 4.15	Flow diagram for data received interrupt .....	73
Figure 4.16	Flow diagram for short address data received .....	75
Figure 4.17	Flow diagram to calculate best offer .....	78
Figure 4.18	Flow diagram for short address timers .....	79
Figure 4.19	Keepalive frame .....	82
Figure 4.20	Request Keepalive frame sample .....	83
Figure 4.21	Acknowledge Keepalive frame sample .....	83
Figure 4.22	Parent-child relationship table view on simulator .....	84
Figure 4.23	Flow diagram for Keepalive received .....	85
Figure 4.24	Flow diagram for Keepalive timer .....	88
Figure 5.1	Connection frame format .....	98
Figure 5.2	Flow diagram for Connection forward .....	102
Figure 5.3	Resolve Medium Access Control address frame .....	104

Figure 5.4	Unicast Resolve Medium Access Control frame sample.....	104
Figure 5.5	Broadcast Resolve Medium Access Control frame sample .....	104
Figure 5.6	Acknowledge Medium Access Control address frame sample.....	104
Figure 5.7	Negative Acknowledge Medium Access Control address frame sample .....	104
Figure 5.8	Flow diagram for Resolve MAC address request .....	106
Figure 5.9	Flow diagram for MAC address timer .....	107
Figure 5.10	Flow diagram for MAC address data received .....	109
Figure 5.11	Request connection frame.....	113
Figure 5.12	Request connection frame sample .....	113
Figure 5.13	Acknowledge connection frame .....	114
Figure 5.14	Acknowledge connection frame sample .....	114
Figure 5.15	Negative acknowledge frame.....	114
Figure 5.16	Negative acknowledge connection frame sample.....	115
Figure 5.17	Remove connection frame .....	115
Figure 5.18	Remove connection frame sample .....	116
Figure 5.19	Flow diagram for Request connection send.....	117
Figure 5.20	Flow diagram for Request connection data received for the coordinator .....	119
Figure 5.21	Flow diagram for Acknowledge connection received .....	120
Figure 5.22	Flow diagram for Negative acknowledge and Remove connection request .....	122
Figure 5.23	Flow diagram for Remove connection request .....	123
Figure 5.24	Flow diagram for connection establish timers .....	125
Figure 5.25	Request endpoint status frame .....	126
Figure 5.26	Request endpoint status frame sample.....	127
Figure 5.27	Acknowledge endpoint status frame sample.....	127
Figure 5.28	Flow diagram for Request endpoint status transmitted.....	129
Figure 5.29	Flow diagram for endpoint status timers .....	131

Figure 5.30	Flow diagram for endpoint status received.....	133
Figure 5.31	Simulator snapshot of coordinator.....	134
Figure 5.32	Simulator snapshot of a connection establish between to endpoints.....	135
Figure 5.33	Snapshot of Device 2 before short address changes.....	144
Figure 5.34	Snapshot of Device 2 after the short address changes.....	144
Figure 5.35	Snapshot of Device 2 after first connection is established.....	148
Figure 5.36	Snapshot of Device 2 after second connection is established.....	150
Figure 5.37	Snapshot of Device 1 after second connection is established.....	150
Figure 5.38	Snapshot of Device 2 after Device 1's endpoint status changes.....	151
Figure 5.39	Snapshot of Device 2 after Device 1's analogue endpoint status changes.....	153
Figure 6.1	Controller basic block diagram requisites.....	156
Figure 6.2	Microchip MPLAB <sup>®</sup> ICD2 programmer.....	157
Figure 6.3	PIC18 Simulator IDE.....	158
Figure 6.4	Micro hardware device setup first phase.....	159
Figure 6.5	Receive USART interrupt routine.....	162
Figure 6.6	Timer interrupt routine for data received.....	163
Figure 6.7	Old connection control field.....	165
Figure 6.8	New connection field.....	165
Figure 6.9	Add connection process modification successful.....	165
Figure 6.10	Add connection process modification unsuccessful.....	167
Figure 6.11	Request connection modified frame sample.....	168
Figure 6.12	Acknowledge coordinator connection frame.....	168
Figure 6.13	Acknowledge coordinator connection sample frame.....	169
Figure 6.14	Request endpoint status modified frame.....	170
Figure 6.15	Request endpoint status modified sample frame.....	170
Figure 6.16	Device circuit diagram.....	172

Figure 6.17	Wireless modem circuit diagram .....	175
Figure 6.18	Radio modem top view .....	176
Figure 6.19	Radio modem bottom view with daughterboard.....	176
Figure 6.20	Device including controller and radio modem.....	177
Figure 6.21	Packets received interlaced.....	177
Figure 6.22	Collision receive byte .....	179
Figure 6.23	Collision transmit packet .....	180
Figure 6.24	Collision timer .....	181
Figure 6.25	Screenshot of the Command Counter window .....	182
Figure 6.26	Screenshot of eight endpoints connected.....	185
Figure 6.27	Relation between devices vs. bits per second .....	188
Figure 6.28	Relation between endpoint connections vs. bits per second .....	190
Figure 6.29	Relation between linear line gradient from Figure 6.28 and devices.....	192

## List of tables

Table 3.1	Short addresses used to forward packet for logical tree.....	52
Table 4.1	Best short address calculation.....	77
Table 4.2	Parent-child relationship table .....	82
Table 4.3	Keepalive time schedule .....	87
Table 4.4	Request short address. Device 1: MAC 0000001641A75694 .....	89
Table 4.5	Request short address. Device 2: MAC 000000065BC31340.....	89
Table 4.6	Request short address. Device 3: MAC 0000000874B38C2E .....	90
Table 4.7	Cleanup parent-child table. Device 1: MAC 0000001641A75694.....	91
Table 4.8	Cleanup parent-child table. Device 3: MAC 0000000874B38C2E.....	91
Table 4.9	New coordinator selected. Device 2: MAC 000000065BC31340.....	92
Table 4.10	New coordinator selected. Device 3: MAC 0000000874B38C2E .....	93
Table 4.11	Two coordinators detected. Device 1: MAC 0000001641A75694.....	95
Table 4.12	Two coordinators detected. Device 2: MAC 000000065BC31340 .....	95
Table 4.13	Two coordinators detected. Device 3: MAC 0000000874B38C2E.....	95
Table 4.14	Force coordinators. Device 1: MAC 0000001641A75694 .....	96
Table 4.15	Force coordinators. Device 2: MAC 000000065BC31340.....	96
Table 4.16	Force coordinators. Device 3: MAC 0000000874B38C2E .....	97
Table 5.1	Endpoint groups.....	110
Table 5.2	Endpoint capabilities.....	110
Table 5.3	Endpoint IO's.....	111
Table 5.4	Device remote endpoint table .....	111
Table 5.5	Coordinator request connection table .....	111
Table 5.6	Unsuccessful connection. Device 1: MAC 0000001641A75694 .....	136
Table 5.7	Unsuccessful connection. Device 2: MAC 000000065BC31340.....	136
Table 5.8	Unsuccessful connection. Device 3: MAC 0000000874B38C2E .....	137

Table 5.9	Successful connection. Device 1: MAC 0000001641A75694.....	138
Table 5.10	Successful connection. Device 2: MAC 000000065BC31340 .....	138
Table 5.11	Successful connection. Device 3: MAC 0000000874B38C2E.....	139
Table 5.12	Short addresses changes. Device 1: MAC 0000001641A75694 .....	141
Table 5.13	Short addresses changes. Device 2: MAC 000000065BC31340 .....	141
Table 5.14	Short addresses changes. Device 3: MAC 0000000874B38C2E .....	142
Table 5.15	Three endpoints. Device 1: MAC 0000001641A75694 .....	145
Table 5.16	Three endpoints. Device 2: MAC 000000065BC31340.....	146
Table 5.17	Three endpoints. Device 3: MAC 0000000874B38C2E .....	147
Table 5.18	Analogue endpoints. Device 1: MAC 0000001641A75694 .....	152
Table 5.19	Analogue endpoints. Device 2: MAC 0000000102A0046D .....	152
Table 5.20	Endpoint deletes connection. Device 1: MAC 0000001641A75694 .....	153
Table 5.21	Endpoint deletes connection. Device 2: MAC 0000000102A0046D .....	153
Table 6.1	3 Wire Straight Cable .....	173
Table 6.2	3 Wire Cross Cable.....	173
Table 6.3	Control point selection.....	173
Table 6.4	Packet count with one device.....	183
Table 6.5	Packet count with two devices, Part 1 .....	184
Table 6.6	Packet count with two devices, Part 2 .....	184
Table 6.7	Packet count with three devices, Part 1 .....	186
Table 6.8	Packet count with three devices, Part 2 .....	187
Table 6.9	Relation between devices vs. bits per second .....	188
Table 6.10	Relation between endpoint connections vs. bits per second .....	190
Table 6.11	Delta gradient between different lines .....	192
Table 6.12	Bandwidth gradient increase with increase in devices.....	192

## Acronyms and abbreviations

<b>Abbreviation</b>	<b>Description</b>
AES	Advance Encryption Standard
AES-CBC-MAC	Advance Encryption Standard Cipher Block Chaining Message Authentication Code
AES-CCM	Advance Encryption Standard Counter with Cipher Block Chaining Message Authentication Code
AES-CTR	Advance Encryption Standard Counter
ASCII	American Standard Code for Information Interchange
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/DA	Carrier Sense Multiple Access with Collision Detection
CCITT	Consultative Committee for International Telegraphy and Telephony
CRC	Cyclic Redundant Check
CTS	Clear to Send
DCE	Data Communication Equipment
DES	Data Encryption Standard
DivX	Digital Video Express
DRAM	Dynamic Random Access Memory
DTE	Data Terminal Equipment.
emf	Electromagnetic field
EEPROM	Electrically Erasable Programmable Read Only Memory
EUI	Extended Organizationally Unique Identifier
FFD	Full Function Device
GHz	Giga Hertz
IEEE	Institute of Electrical & Electronics Engineers

Kb/s	kilobit per second
LED	Light Emitting Diode
MAC	Medium Access Control
MHz	Mega Hertz
MIC	Message Integrity Code
MP3	MPEG Audio Layer 3
MP4	Moving Picture Experts Group 4 (also called MPEG4)
MPEG	Moving Picture Experts Group
OSI	Open Systems Interconnection
PAN	Personal Area Network
PANID	Personal Area Network Identifier
PC	Personal Computer
RAM	Random Access Memory
RFD	Reduced Function Device
RTS	Request to Send
ROM	Read Only Memory
TCP	Transmission Control Protocol
TTL	Time To Live
IC	Integrated Circuit
ID	Identification
IO	Input Output
IP	Internet Protocol
ISO	International Standards Organization
UDP	User Datagram Protocol
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
WLAN	Wireless Local Area Network

WMA	Windows Media Format
WPAN	Wireless Personal Area Network
XOR	Exclusive-OR

# Chapter 1

## 1 Introduction

The San Francisco Chronicle stated in 2005 that wireless mesh networks would in future transform our world. Nodes inside sensors send streams of messages to each other that allow us to control and monitor the environments both at home and at work [1].

Major companies agreed to set hardware and software standards called Zigbee. “Although ZigBee allows nodes to communicate, some experts don't consider it to be a true mesh network because the initial deployment happens to require some plugged-in devices” [1]. Another problem with the Zigbee protocol is that “for many applications, Zigbee networking functionality is overkill” [2].

In this dissertation, alternative methods to control wireless devices will be investigated and based on the Institute of Electrical and Electronics Engineers (IEEE) 802.11.4 standard. The IEEE 802.11.4 standard specifies the physical and medium access control layer for a wireless personal area network (WPAN) [3]. It is the basis for the Zigbee and MiFi specification, which offers a complete network solution [4].

The aim of the study was a microprocessor controlled mobile device triggering a certain action on a receiving device in a wireless network. The two devices do not need to communicate directly with each other. The device must be able to discover and locate neighbour devices and negotiate a master controller in a basic secure method.

In this paper, the emphasis is placed on the development and evaluation of such a protocol, and the microcontroller implementation is part of the next phase of the research.

## 1.1 Hypothetical solution

A group of devices can share their information inexpensively, if the transmission distance is limited and if all devices do not communicate with each other. This implies that:

- software will be designed to allow devices to discover each other;
- a non-controlled frequency spectrum can be used to transmit sensitive data;
- a certain condition on a device can trigger an output on another device;
- the only human intervention needed must be when relationships are established between devices;
- the algorithm must be able to operate on a basic microprocessor, with limited capabilities or intelligence.

## 1.2 Expected outcomes

The system will be tested with a building monitoring system because it is static and a less expensive choice making the evaluation more suitable. An analyser must first be developed to simulate and test the algorithm on a personal computer. Finally the algorithm must be loaded on an eight-bit microprocessor. The algorithm must be tested on a system with at least three wireless devices and the other devices can be simulated on an Ethernet connected computer network.

Figure 1.1 illustrates a typical home environment connected as a network. The evaluation setup consists of lights, a gate motor and alarm system that will be monitored. Each system must have one master device 'A', and any device must have the capability to become a master device if the current master device fails. All devices, including the master device, can monitor inputs and set or reset outputs. Inputs must be on/off conditions or variable inputs like temperature and day/night sensors.

Associations between inputs and outputs can be one-to-one and many-to-one. A device must be capable to host multiple endpoints.

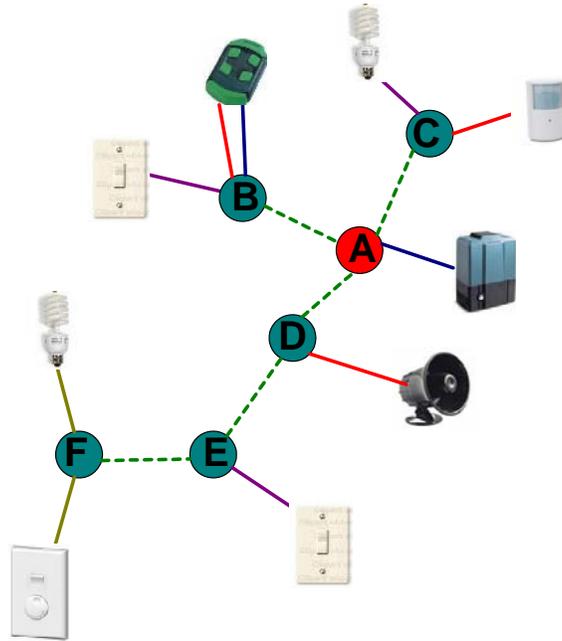


Figure 1.1 Typical home environment connected as a network

Intrusion detection is an example of a many-to-one association. Device B is connected to a remote control button #1, C to a motion sensor and D connected to a siren. When any of the inputs are activated the siren is triggered.

Device A is connected to the gate motor and Device B is connected to the remote control button #2. The automated gate motor is an example of a one-to-one association.

For the room lighting, Device F is connected to a dimmer switch and a light. This is an example of a one-to-one association, but the endpoints can have variable input values.

For the passageway, Device B and E is connected to light switches and Device F to a light bulb. When any of the inputs are activated the light bulb will switch on. This is an example of a many-to-one association.

### **1.3 Layout of dissertation**

The dissertation is set out as follows:

- Chapter 2

Different communication principles and techniques are discussed on how to find a solution for the problem stated.

- Chapter 3

A proposed overview of the solution is discussed. All the commands the protocol necessitates is briefly discussed and grouped together in logical layers.

- Chapter 4

A detailed discussion of the lower two layers namely the device discovery and maintenance. The simulation software is implemented to test this layer.

- Chapter 5

The simulator software is developed and tested for the upper two layers namely: Maintaining connections between endpoints.

- Chapter 6

The microcontroller software and hardware is developed and tested. Tests are done with the simulators and hardware simultaneously.

- Chapter 7

The conclusion of the dissertation.

# Chapter 2

## 2 Communication principles

Chapter 2 covers the basic communication principles that were needed to develop a method to safely synchronise information in a wireless mesh network.

### 2.1 Open Systems Interconnection Reference Model

The International Standards Organization (ISO) released a set of specifications called the Open Systems Interconnection (OSI) reference model to standardise network communications. The model divides network communications into seven layers, shown in Figure 2.1 [5][6].

7	Application		Upper Layers
6	Presentation		
5	Session		
4	Transport		
3	Network		Lower Layers
2	Data Link		
1	Physical		

Figure 2.1 Seven-Layer OSI model

When a device is sending data, each layer receives the data from the above layer. Each layer performs an action and prepares the data for the following layer by adding its own information, called a header, in front of the data. The last layer, the Physical layer, delivers the data to the physical communication

media such as a network cable. The opposite happens at the receiving device [5]. Each layer is setup to look like it is communicating directly to the same layer on the other device as shown in Figure 2.2 [6].

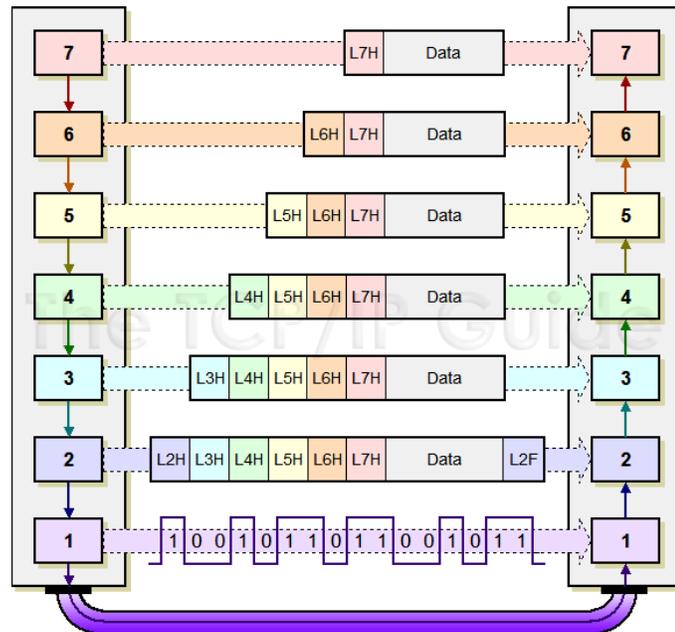


Figure 2.2 OSI relationship

### 2.1.1 Physical layer

The Physical layer sends and receives bits. A bit can only be a '0' or a '1' [7]. This layer also defines the electrical and the physical specifications of the medium used for communication [5].

### 2.1.2 Data Link layer

The Data Link layer prepares the data received from the upper layers for transmission. The physical device address, the Medium Access Control (MAC) address, uniquely identifies each device in a network. If a device is not on the local network the MAC address of the nearest router is used. Finally the layer manages error correction [5].

### **2.1.3 Network layer**

The Network layer is one of the most complex layers and manages the addressing in a complex network. The address scheme is logical and hierarchical [6]. Networks are joined with devices called routers. Routers use routing tables and algorithms to determine how to send information to its destination. Each network is assigned a network address range and every device on a network is assigned a network address [5]. Routers must continually inspect the network to determine the best path to a destination device [7]. The traffic is transferred connectionless one hop at a time. It is not responsible for reliability; however, it is responsible for the detection of errors in order for them to be discarded [6].

### **2.1.4 Transport layer**

The Transport layer ensures reliable delivery of data between devices through flow control, error control, segmentation and sequencing. Some applications use connection-oriented services and the Transport layer can request retransmissions or acknowledge data received [5].

### **2.1.5 Session layer**

The Session layer establishes, manages and terminates communication between the local and remote devices. Three types of dialogues are used:

- Simplex - Data only flows in one direction.
- Half duplex -Data flows in both directions, but only in one direction at a time.
- Full duplex - Data flows in both directions simultaneously [5].

### **2.1.6 Presentation**

The Presentation layer receives and formats data from the Application layer and if encryption is used, it encrypts the data into the correct format, before presenting it to the Session Layer [5][6].

## 2.1.7 Application

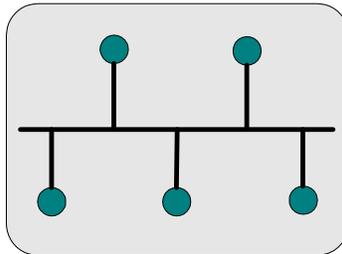
The Application layer is not the application running on the computer, but it interacts with software that needs to communicate with another device [5].

## 2.2 Network communication fundamentals

### 2.2.1 Network topologies

A topology is the physical layout of devices connected to each other. Topologies can be physical or logical. A physical topology refers to how the devices connect to each other and a logical topology refers to how the data flows. Some of the main topologies are:

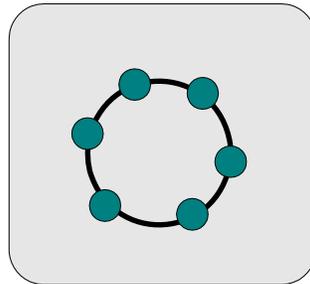
- Bus topology



*Figure 2.3 Bus topology*

All devices in a network are connected with a single cable. This is one of the easiest and cheapest topologies to install. Both ends of the cable must be terminated or else the signal will bounce back resulting in noise on the network. Only one device can transmit data and the other devices must listen to all the data and will only accept data addressed to them. When there is a cable break the entire network is down [5].

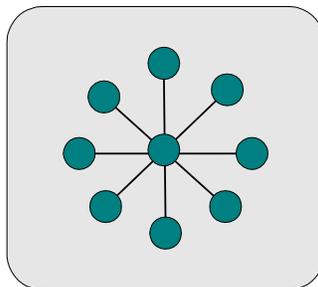
- Ring topology



*Figure 2.4 Ring topology*

Each device is connected to the device next to it and the first and last device is connected forming a ring. When data is transmitted each device must inspect it. If the data is destined for the receiving device it is retrieved, otherwise it is retransmitted on the ring to the next device. There is no single point of failure, if the cable breaks between two devices, the data can be transmitted in the other direction. This topology makes it difficult to add a new device and it is more expensive to implement than other topologies [5].

- Star topology

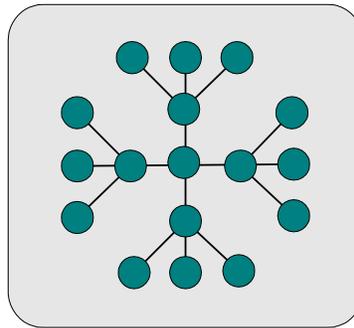


*Figure 2.5 Star topology*

The Star topology is one of the most popular topologies. This topology is easy to setup and relatively cheap, but more expensive than the bus topology because there is more cables needed. Each device is connected to a central device. This topology is more redundant than

the bus topology, if there is a cable break one device is affected. The only single point of failure is the central device [3].

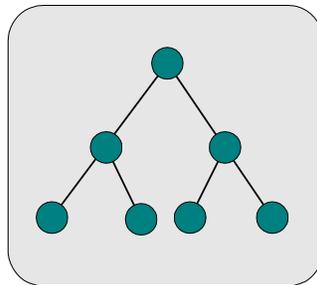
- Extended Star topology



*Figure 2.6 Extended Star*

The Extended Star topology is more advanced than the Star topology. All devices connect to a sub-central device and they connect to the central device. This creates more single points of failure, but it is more scalable and is generally used in large networks.

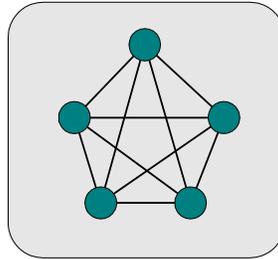
- Hierarchical topology



*Figure 2.7 Hierarchical topology*

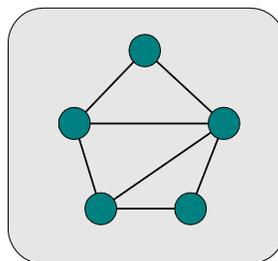
The Hierarchical topology is also referred to as the Tree topology. This topology is almost the same as the Star topology, since every device connects to the central device. If the top device fails the complete network will be down.

- Mesh topology



*Figure 2.8 Fully Mesh topology*

In a Fully Meshed topology, every device is connected making it a truly redundant network. This solution is very expensive.



*Figure 2.9 Partial Mesh topology*

A Partial Mesh topology is almost the same as a Fully Meshed network, but each device has at least one alternative link.

## **2.2.2 Collisions**

When a shared media topology is used, such as Ethernet and Wireless, it is possible to have collisions when two or more devices are trying to transmit data at the same time [5]. There are two methods to manage collisions:

- *Carrier Sense Multiple Access with Collision Detection (CSMA/CD)*

CSMA/CD can detect if a collision occurred. The device will 'listen' to see if another device is busy transmitting, and if not, start to retransmit. It will keep listening while transmitting because two devices could start transmitting at the same time. If a collision is detected, a signal is sent out to inform all devices to stop transmitting. All devices will stop transmitting before it will start the process over. CSMA/CD does not prevent collisions, but only helps managing collisions [5].

- *Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)*

CSMA/CA is used on wireless networks where collision detection is not possible. It can optionally exchange Request to Send (RTS) by the sender and Clear to Send (CTS) by the receiver, alerting all receivers in range to stop transmitting for a random duration [1][5].

### **2.2.3 Communication methods**

There are two types of communications, namely connection-oriented and connectionless. Both types are operating in the fourth layer, the Transmission Layer [7], of the OSI model.

- *Connection-oriented*

It ensures reliable data delivery between two devices and can handle errors without the help of the upper layer. The sender starts to establish a session, with some kind of handshaking, with the other device [5]. One of the most common protocols today is Transmission Control Protocol/Internet Protocol (TCP/IP). It is used on networks that are not very reliable. Port numbers are used to keep track of different conversations over the same network [7].

- *Connectionless*

The transmitting device does not require the receiving device to notify on data received and is therefore unreliable. User Datagram Protocol (UDP) is an example of connectionless communication in the IP stack and also uses ports to differentiate between applications. It does not use as many resources as TCP. Monitoring data and real time traffic like Voice and Video uses connectionless communication [7].

## **2.3 Integrity of data**

Data can be changed accidentally or deliberately. Data is changed accidentally when the transmission medium is faulty and Cyclic Redundancy Check (CRC) is used to identify corrupted data packets. Hackers change data deliberately to access information or to corrupt a system and encryption is used to protect data against this type of attack.

- Cyclic Redundancy Check

Cyclic Redundancy Check (CRC) is a non-secure algorithm to detect accidental changes in raw data. CRC is used when data is stored or when data is sent over transmission mediums. A fixed length CRC checksum is calculated for each block of data. The sender will append CRC checksum without changing the data. The receiver recalculates the CRC checksum and compares its CRC checksum with the sender's CRC checksum. If the two values do not match, the block of data contains a data error and the device can now take corrective actions [10].

The calculation for the CRC checksum involves a division of the data by the specific system generated polynomial. The rest of the division is the CRC checksum. The mathematical method to calculate the CRC code is standard, but the polynomial used is not. For CRC-16

the polynomial is  $G(x) = x^{16} + x^{15} + x^2 + 1$  and for CRC-16-CCITT it is  $G(x) = x^{16} + x^{12} + x^5 + 1$  [8]. CRC-1 up to CRC-256 is available today. The number is an indication of the length of the CRC checksum in bits [10]. The polynomial is a representation of an eight-bit binary word and CRC-16-CCITT can be translated to 1000000000000101b, x16 is not represented but used in the calculations [8].

- Encryption

A symmetric cryptographic algorithm uses the same key to encrypt and decrypt data. There are two types of cryptographic algorithms, namely stream ciphers and block ciphers.

*Stream ciphers*

The message is represented in a binary stream. A key is Exclusive-OR (XOR) to the plain text to generate a cipher text as illustrated below [11]:

Encryption

Plain text: 01011010b

Key: 00010010b

Cipher text: 01001000b

Decryption

Cipher text: 01001000b

Key: 00010010b

Plain text: 01011010b

### *Block ciphers*

Encryption and decryption is applied to a fixed block of data. Data Encryption Standard (DES) and Advance Encryption Standard (AES) are examples of block ciphers [11].

#### 1. Data Encryption Standard

It is the first government-approved encryption standard that was released for public use. It operates in blocks of 64-bits plain text resulting in 64-bits cipher text. The data is padded at the end if it is not 64-bits. The key is 56-bits including 8-bit parity.

Triple DES is the DES algorithm applied three times to each data block, typical with two 56-bit keys, but today it is classified as unsecured due to the short key size [11].

#### 2. Advance Encryption Standard

In 2000, AES Rijndael becomes the official DES replacement. It is a block cipher that can be used with 128-bit data blocks with 128-, 192- and 256-bit key length [11]. The AES is secure and fast because its algorithm is based on XOR operations and bit shifts. This makes it possible to perform extremely fast AES calculations in hardware.

An incomplete block is filled through a process called padding starting with the characters '0'. The algorithm involves four mathematical processes: SubBytes, ShiftRow, MixColumn and AddRoundKey. The input to each round is called the state. With the SubBytes process each byte is replaced using a nonlinear S-box also known as an Encryption substitution table [11].

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

A	B	C	D
F	G	H	E
K	L	I	J
P	M	N	O

Figure 2.10 AES ShiftRow process

The ShiftRow step executes a cyclical, left shift for each state except for the first row. The second row is shifted once, the third row twice and the fourth row is shifted three times. This applies to 16- and 24-byte blocks as illustrated in Figure 2.10. The third row in a 32-byte block is shifted four times [9].

The MixColumn process involves the multiplication of each column  $a_i$  of the state by a fixed matrix  $c(x)$  following polynomials with coefficient  $GF(2^8)$  as illustrated in Figure 2.11 [9].

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Figure 2.11 AES MixColumn process

Lastly in the AddRoundKey the next round's key is XOR with the state.

## 2.4 Protocols for Wireless Personal Area Networks

### 2.4.1 IEEE 802.15.4

This is a standard that specifies the physical and medium access control layer for low rate Wireless Personal Area Networks (WPAN) and involves little or no infrastructure. This standard defines only

two network layers and is based on the two lower layers of the OSI model. It is the basis for the MiWi and ZigBee protocols, which offers a complete network solution as illustrated in Figure 2.12.

The two layers are called the Physical and Medium access control layers [3]:

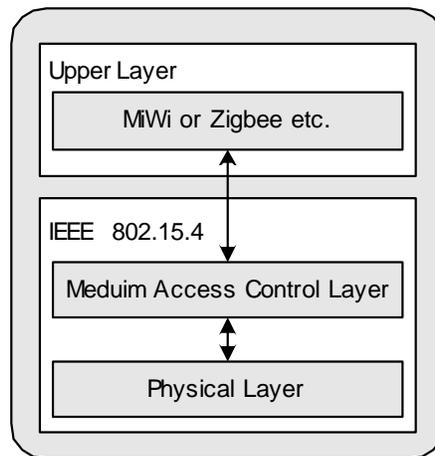


Figure 2.12 IEEE 802.15.4 Protocol stack

- *Physical layer*

The Physical layer is responsible for managing the radio transceiver, energy detection, CSMA/CD, data transmission and channel selection [2]. The standard specifies three unlicensed frequency bands:

1. 868-868.8 mega hertz (MHz): Europe, allows one communication channel.
2. 902-928 MHz: North America, allows up to thirty channels.
3. 2400-2483 MHz: Worldwide allows sixteen channels [3].

Transfer rates 20, 40, 100 and 250 kilobit per second (kb/s) are allowed for the first two frequency bands and 2450 MHz only supports 250 kb/s [3]. If lower power consumption is needed the transfer rate can be reduced. This layer is responsible for channel selection and energy management.

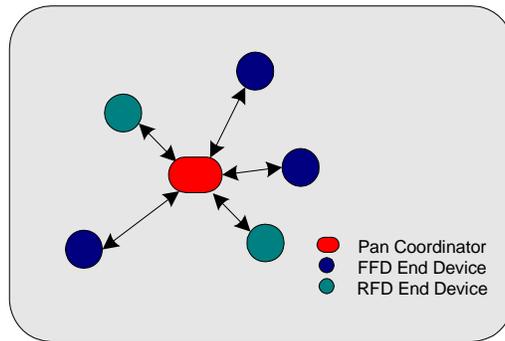
- *Medium Access Control layer*

The Medium Access Control (MAC) layer is responsible for managing all access to the physical radio channel, generate and manage network beacons, supporting device security and providing a reliable link between two MAC entities [3].

Two network devices are defined: Full Function Device (FFD) and Reduce Function Device (RFD). The FFD can operate as a Personal Area Network (PAN) coordinator, coordinator or end device. FFD can communicate with other FFD's and RFD's. A RFD can only be an end device and can only link to one FFD [3].

The IEEE 802.15.4 standard supports two topologies, namely the Star and Peer-to-Peer topology. In either topology each device must have a unique 64-bit address. The PAN coordinator can optionally associate a 16-bit short address with every device [3].

- *Star topology*



*Figure 2.13 IEEE 802.15.4 Star topology*

In a Star topology several devices communicate via a single PAN coordinator. After the first FFD is activated, it establishes its own network and becomes the PAN coordinator. Each Star network, within the radio sphere of influence, must have a unique PAN identifier. Once the

PAN coordinator has chosen a PAN identifier, other FFD's and RFD's can join its network as illustrated in Figure 2.13 [3].

- Peer-to-Peer topology

In a Peer-to-Peer topology each FFD is capable of communication with any other device. One FFD is nominated as the PAN coordinator and RFD's can only connect to one FFD [3].

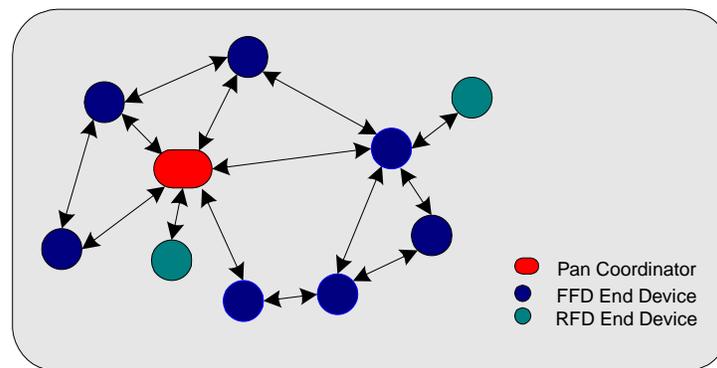


Figure 2.14 IEEE 802.15.4 Peer-to-Peer topology

The cluster tree is an example of the Peer-to-Peer topology where most devices are FFD's. An RFD will connect as the leaf device at the end of a branch. Any FFD can act as a coordinator and only one of them can be the PAN coordinator. The PAN coordinator forms the first cluster by choosing an unused PAN identifier. Other devices gradually connect and form a multi cluster network with parent-child relationships between devices as illustrated in Figure 2.15 [3].

The 802.15.4 standard defines the additional use of a super frame structure, defined by the coordinator and is bounded by network beacons sent by the coordinator. The super frame is divided into 16 equal size slots and can optionally have an active and inactive period, during which the coordinator may enter a low-power mode. The beacon is sent in the first slot and is used to synchronise attached

devices, to identify the PAN and to define the super frame structure. If a device needs to communicate between beacons, it must use the CSMA/CA method [3].

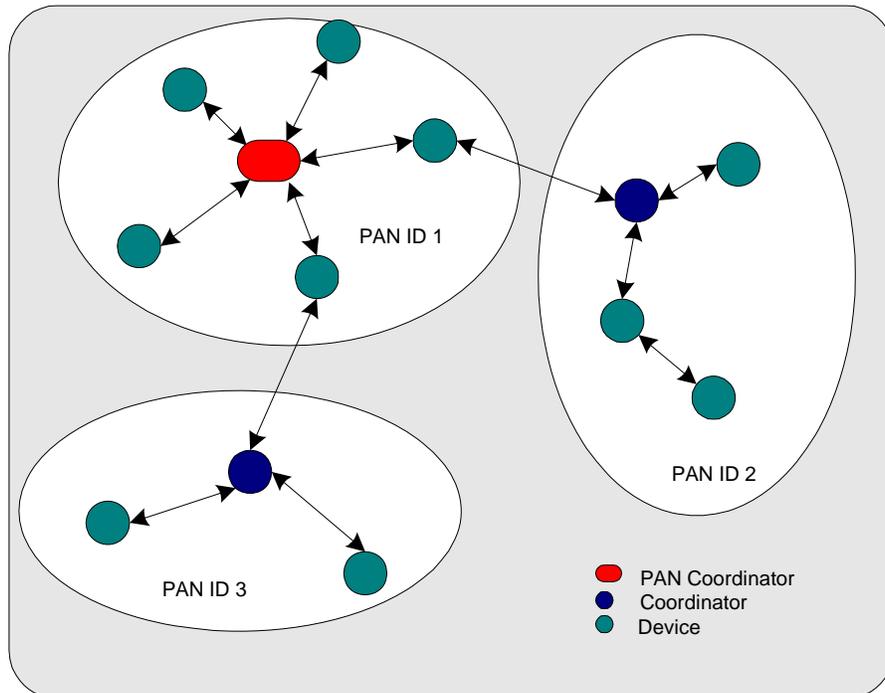


Figure 2.15 IEEE 802.15.4 Multi Cluster Network

The frame structure has been designed to transfer data simply. Currently four types are defined:

- Beacon frame, used by coordinators to send beacons.
- Data frame used for all data transfers.
- Acknowledgement frames used optionally to confirm frame reception.
- MAC frames used for all MAC control transfers [3].

Any wireless network is vulnerable to tampering and attacks. The IEEE 802.15.4 standard has limited security because the devices are low cost, with limited processing power, storage and power sources. The cryptographic algorithm is based on symmetric-key cryptography and uses keys that are provided

by the upper layers, to protect the payload. The key can be shared between two devices or a group of devices [3].

## 2.4.2 MiWi™ Wireless Network Protocol Stack

The MiWi™ protocol was designed for low data rates, short distance and low cost networks and fundamentally based on the IEEE 802.15.4 standard. It provides an easy alternative for smaller networks with few hops between nodes [12]. Three types are defined according to their function: Personal Area Network (PAN) coordinator, coordinator and end device.

- *PAN coordinator*

There can only be one PAN coordinator. It starts the network, selects the channel and PANID, allocates addresses and holds binding tables. Any device joining a PAN must use the PAN coordinator's settings. It can be compared with the IEEE FFD [12].

- *Coordinator*

There can be a maximum of eight coordinators that can handle 127 children. It extends the range of the network allowing more nodes per network and can also perform monitoring and control functions. It compares with the IEEE FFD [12].

- *End device*

The end device can only perform monitoring and control functions. It can be compared with the IEEE FFD and RFD [12].

Packets can travel a maximum of four hops and two hops from the PAN coordinator. It is capable of having 1024 nodes on a network [12]. MiFi supports three types of topologies:

- *Star*

It consists of one PAN coordinator and one or more end devices. No end device can transfer data directly to another end device, it can happen only via the PAN coordinator [12].

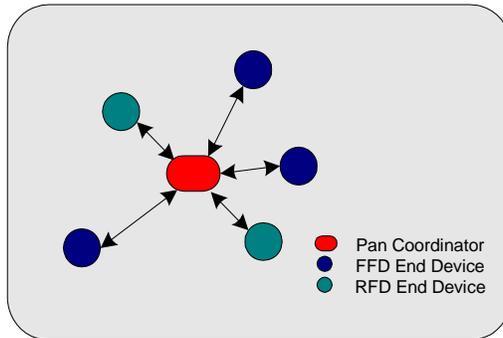


Figure 2.16 MiWi Star topology

- *Cluster Tree*

Here is also only one Pan coordinator, but multiple coordinators and end devices can connect to the PAN coordinator. Only End Devices connects to coordinators. Data flow will always follow the tree format and will go through a PAN coordinator or coordinator. Sometimes the coordinators must route traffic to another coordinator [12].

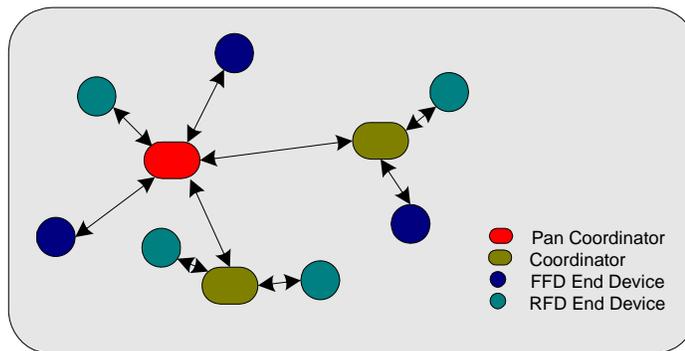


Figure 2.17 MiWi Cluster Tree topology

- *Mesh*

It is similar to a Tree topology apart from the fact that an FFD can route traffic to any other FFD. The advantage is that network latency is reduced and reliability increased [12].

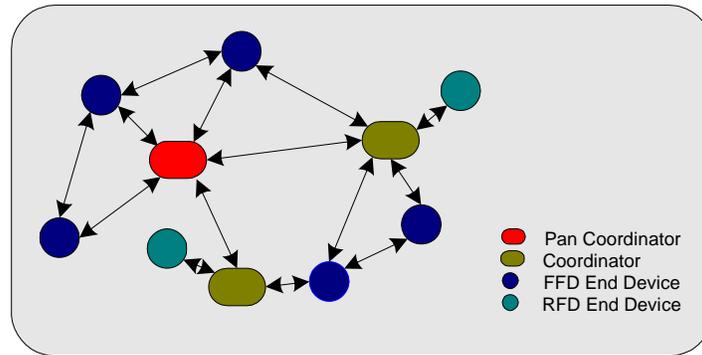


Figure 2.18 MiWi Mesh topology

MiWi protocol supports the IEEE 802.15.4 address structure and uses the following addresses:

- *Extended Organizationally Unique Identifier (EUI)*

This address must be unique to any IEEE 802.15.4 device and is an eight-byte number. The upper three bytes must be purchased from the IEEE and the lower five bytes must be allocated by the company to make every device EUI unique.

- *PAN Identifier (PANID)*

All nodes in a PAN must share the same two-byte address. A device assumes the PANID when it wants to join a PAN.

- *Short address*

This is a two-byte unique address assigned to a device by its partner. The IEEE specifies the PAN coordinator short address must always be 0000h. The parent's number fields are bits eight to ten. This is why there can only be eight coordinators and one PAN coordinator. The

child number fields are bits zero to six and a coordinator will always be 0000h. These addresses are used to communicate with other devices in the network. Figure 2.19 illustrates how the short addresses are allocated [12].

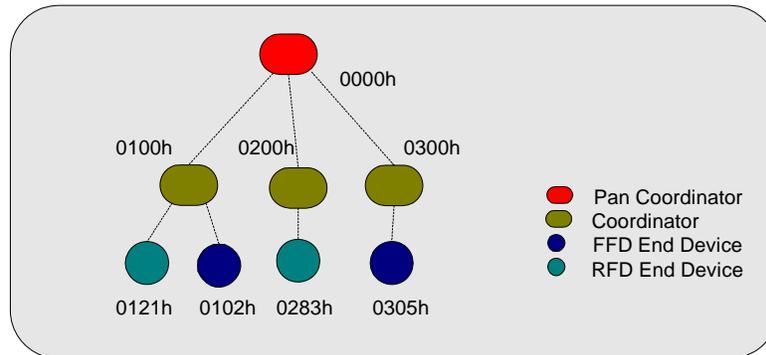


Figure 2.19 MiWi short address allocation

Routing in a wireless network can be difficult and resource intensive. One of the routing algorithms is to determine to who the next hop is for an outgoing packet. When a device joins a network, a beacon is sent out to all coordinators in range informing them of its information. The coordinators forward this information to their neighbouring devices. Routing becomes easy once all neighbouring devices know about all this information [12].

MiWi uses reports to send information between devices. There can be 256 types of reports but the first report, zero, is reserved for the MiWi Protocol Stack [12].

A MiWi security mode uses Advance Encryption Standard (AES) and can be divided into three areas:

- *Advance Encryption Standard-Counter (AES-CTR)*

This mode encrypts only the payload. An attacker will need a key to understand the message.

It cannot verify frame integrity or the source address.

- *Advance Encryption Standard-Cipher Block Chaining-Message Authentication Code (AES-CBC-MAC)*

This mode ensures frame integrity of the packet. A Message Integrity Code (MIC) is attached to the packet insuring that the packet including the header was not tampered with.

- *Advance Encryption Standard Counter with Cipher Block Chaining Message Authentication Code (AES-CCM)*

Combines AES-CTR and AES-CBC-MAC to ensure both the integrity and encryption of the packet [12].

### **2.4.3 Zigbee**

It is an open standard for wireless networks, developed by the ZigBee Alliance for low data rate networks. It is free for the general public for non-commercial use, but an annual membership fee must be paid to the Zigbee Alliance for commercial use [2][4]. Zigbee needs to meet the following requirements:

- low cost
- ultra-low power
- unlicensed radio bands
- cheap and easy installation
- flexible and extendable networks
- integrated intelligence for network setup [4][13].

Typical application areas are:

- *Building automation*

Examples of applications that can be automated are air-conditioning, lighting, irrigation and safety, including intruder and fire detection.

- *Healthcare*

Wireless sensors monitoring a patient, for example use in fitness programs.

- *Vehicle monitoring*

Vehicles use sensors to monitor devices; a good example is to measure tyre pressure where cables are not possible.

- *Agriculture*

Help farmers to monitor their properties' environmental conditions. Message can be relayed via multiple nodes [4][13].

The ZigBee adopted the IEEE 802.15.4 Physical and Data layer specifications and also operates in the unlicensed industrial, scientific and medical radio bands; 868 MHz, 915 MHz and 2.4 giga hertz (GHz) [2]. Figure 2.20 shows the different channels and speeds for each frequency band.

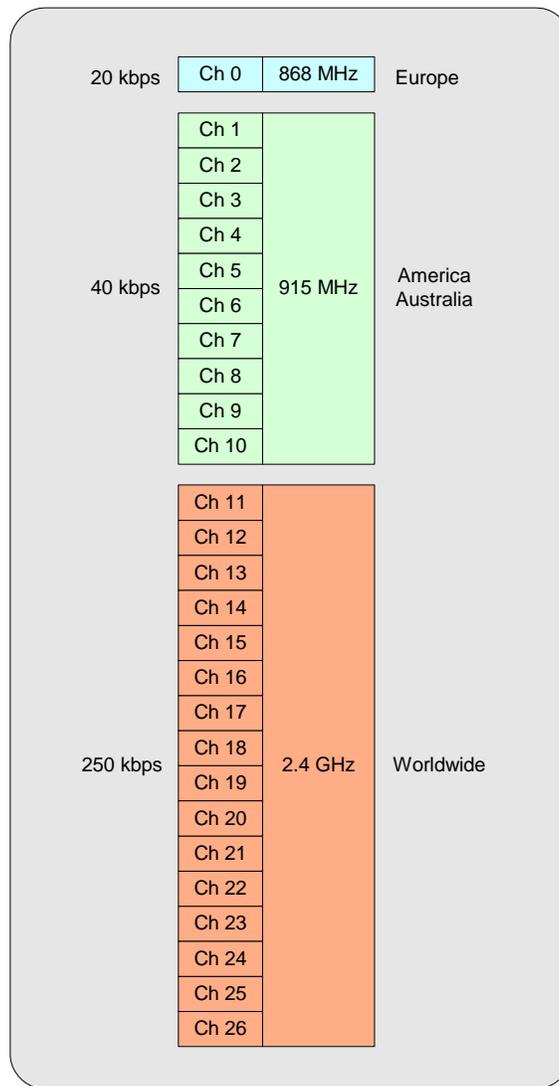


Figure 2.20 Zigbee Frequency Bands

ZigBee supports three node types in the Star, Tree and Mesh topology [2]:

- *Coordinator*

Each network can only have one coordinator and it is the node that starts the network originally. After the coordinator selects the least active frequency channel it assigns a pre-determined or dynamically PANID by detecting if other networks PANID's are operating in the same frequency channel and choosing a PANID that does not conflict with theirs. Other

nodes are now allowed to join this network. It also stores information about the network including the security keys [2].

- *Router*

A router is a node that has the capability to forward messages over the best path. In a Star topology the coordinator can handle these functions, but in a Tree or Mesh topology, at least one router node is needed. Routers can act as end devices if needed, but cannot go into sleep mode to save power [2].

- *End device*

End devices can only send and receive messages to parent nodes, which are either a coordinator or a router. End devices require the least amount of memory and are therefore the cheapest nodes to manufacture. If the node does not need to send or receive messages it can hibernate to conserve power since it is usually battery powered [2].

Figure 2.21 to Figure 2.23 demonstrates the three node types in the different topologies.

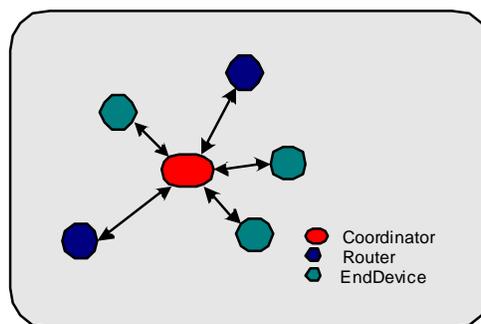


Figure 2.21 Zigbee Star topology

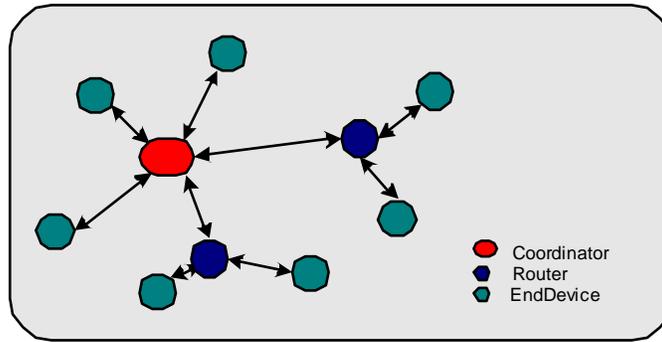


Figure 2.22 Zigbee Tree topology

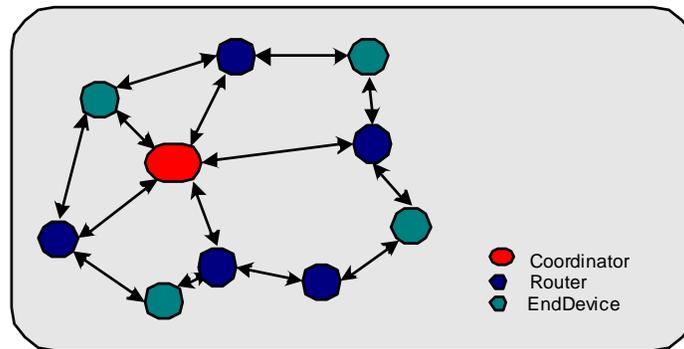


Figure 2.23 Zigbee Mesh topology

The Zigbee protocol supports the IEEE 802.15.4 address structure, which uses two addresses:

- *IEEE address*

This address must be unique to any other IEEE 802.15.4 device in the world and is an eight-byte number. It is also called the MAC address or the extended address.

- *Network address*

This is a two-byte number to uniquely address a node in a local network. The parent router or coordinator node allocates addresses when a node joins a network. The coordinator always reserves the network address 0x0000 [13] for itself.

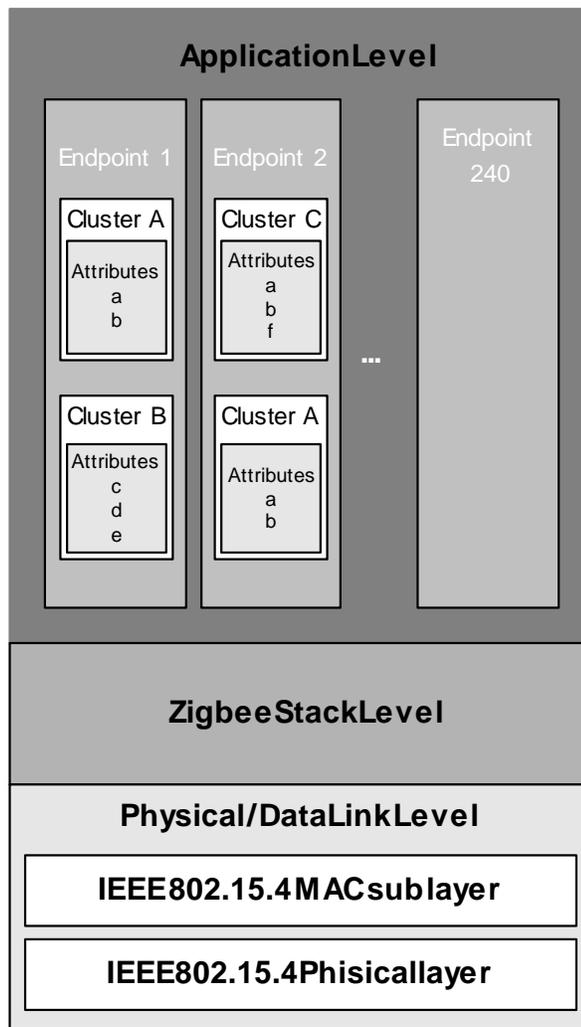


Figure 2.24 Zigbee OSI model

The Zigbee software stack has three basic levels as illustrated in Figure 2.24.

- *Application level*

The application gives the device its functionality. It is where the input is converted into digital data and converted back into an output. Endpoints make it possible for the Application level to run different applications on the same node. Endpoints are numbered from 1 to 240. If the device for example is an environmental sensor, the temperature, humidity and atmospheric

pressure sensors will each be an endpoint address. Endpoint address 0 on each node is reserved to define the node type: coordinator, router and end device. Endpoint address 255 is used to broadcast messages to all endpoints in a node.

Each application, such as home control lighting or security, has its own Application profile that is defined by the Zigbee Alliance. Each device in an Application profile will have a Device Identifier defining the device description. Within the home control lighting profile, devices such as a light sensor, dimmer, occupancy sensor will have a Device Identifier associated with it. Attributes are part of the device description defining the format of the message such as 8-bit, 16-bit and time values. It also defines if it is an input or output. A profile can support more than one attribute. Attributes are grouped together into a cluster [13].

- *Zigbee Stack level*

The Zigbee Stack level is the interface between the Application and Physical and Data Link level. It handles the network structure, routing and security [13].

- *Physical and Data Link level*

The Physical and Data Link level is concerned about the addressing and message transmission and reception. It is based on the IEEE 802.15.4 standard [13].

Two applications can be bound together if they have compatible clusters. Bindings are stored locally or on the coordinator in binding tables. Bindings can be one-to-one, one-to-many and many-to-one [13]. To ensure reliable communication the following measures must be in place:

- CSMA-CA,
- acknowledgements and
- alternative routes.

To prevent hostile intrusion 128-bit AES encryption can be used [13].

## **2.5 Summary**

The discussed topologies and protocols will be used to find an optimal solution to control multiple controllers in a wireless environment.

## Chapter 3

### 3 Standard development of a secure synchronisation of an information system in a wireless mesh network

Chapter 3 denotes the process of the development of a secure synchronisation of an information system in a wireless mesh network.

#### 3.1 Basic understanding

##### 3.1.1 Requirements

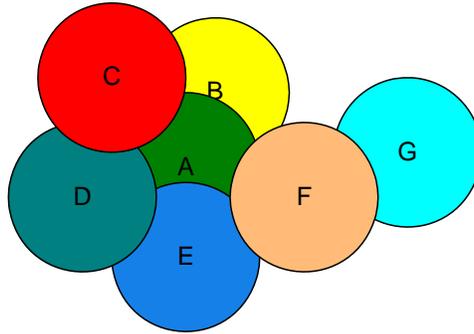
The protocol must be able to do the following:

- Discover a manageable logical topology from a mesh topology.
- Devices must be able to discover the network automatically, move around and recover from power failures.
- No single point of failure.
- Send packets to the correct destination.
- Only controls with the same characteristics can establish a relationship with each other, for example a light switch can only control lights and not an alarm system.
- Must be hacker and intrusion free.

##### 3.1.2 Discover manageable logical topology from a Mesh topology

When devices are connected via a wire it is easy to determine the topology, because it is the same as the physical layout like a Bus, Star, or Ring topology to name a few. Wired devices can only send a packet to the wired receiver, but when sending it wireless, it is not the same thing. Wireless devices

physically form a Star network when one base unit is used, but when the devices are setup to use peer-to-peer communication, the topology changes to a Mesh network. Figure 3.1 illustrates devices A-G reception with perfect sphere antennas.



*Figure 3.1 Wireless topology shown as antenna connectivity with perfect sphere*

From Figure 3.1 we can construe the following assumptions:

- Device A can communicate with B, C, D, E and F,
- Device B can communicate with A, C and F,
- Device C can communicate with A, B and D,
- Device D can communicate with A, C and E,
- Device E can communicate with A, D and F,
- Device F can communicate with A, B, E and G,
- Device G can communicate with F.

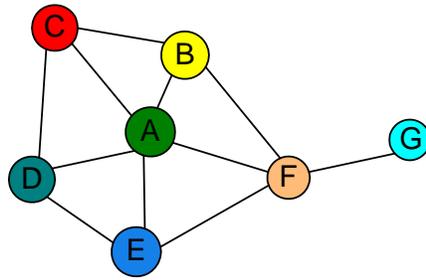


Figure 3.2 Wireless topology shown with lines

Figure 3.2 is the same diagram as Figure 3.1, it only makes use of lines to illustrate connectivity. In section 2.4 we learnt that IEEE 802.15.4 uses only one coordinator and that both Zigbee and MiWi protocols adopted this standard. In section 2.4.2 we also learnt about the use of short addresses. One of the new WPAN protocol standard is that there cannot be a single point of failure and that the devices must discover the network without any additional configuration. To manage a Mesh network is complex and difficult, therefore address allocations will be used to change the physical Mesh network into a logical Star topology.

#### *Short addresses*

Short addresses will also be used, but without the limitation of only ten parent devices as discussed in section 2.4.2 with the MiFi protocol. Two bytes will also be reserved for the short addresses but will be allocated differently.

Figure 3.3 illustrates how the short address structure will work. If the short address is written in hexadecimal it will be 0xABCD. The first digit represents the Personal Area Network (PAN) number. We will be able to connect 16 PAN's together. This is out of the scope of this research and is reserved for further research. For our investigation 'A' will always be a zero. The first device on the network will create the network and choose short address 0x0000 and become the coordinator and the parent for the first fourteen devices in range of the coordinator.

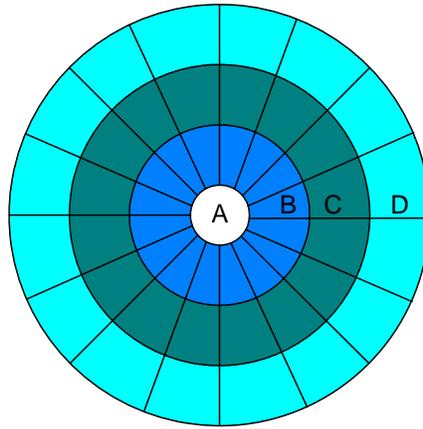


Figure 3.3 Short address structure

When the second device joins the network, the coordinator will accept its first child and give out address 0x0100. This will continue until address 0x0E00 is given out. When the next device requests a short address the coordinator will not offer an address and it will only get an offer from the device with an address 0x0X00. The device will be offered an address of 0x0X10. The second device with the same parent will receive an address of 0x0X20, but when a device is not in range of parent 0x0X00 and only with 0x0Y00, it receives an address of 0x0Y10. If another device connects and is in range of parent 0x0X00 and 0x0Y00 it will receive an offer of 0x0X30 and 0x0Y10. The requester will accept the address nearest to the coordinator if more than one parent exists. If X=1 and Y=2 the child address will be 0x0130.

If a device is only in range of device 0x0XA0 it will get an address 0x0XA1. If the device is in range of parent 0x0X00 and 0x0YA0 and X=6, Y=1 and A=1 address 0610h and 0111h would be offered but only 0610h would be accepted because it is closer to the coordinator.

The child will first look for the offer nearest to the coordinator and then the smallest.

Address 0x0 and 0xF are reserved for parent and broadcast respectively. This means that each parent can have a maximum of 14 children. The maximum devices in a PAN can be calculated as follows:

$$\begin{aligned}
 \text{Maximum devices per PAN} &= \text{Sum of maximum devices per ring} \\
 &= \sum_{n=0}^3 14^n \\
 &= 2955 \text{ devices}
 \end{aligned}$$

This is only a mathematical value and does not take any external factors such as radio transmission speed and interference into account.

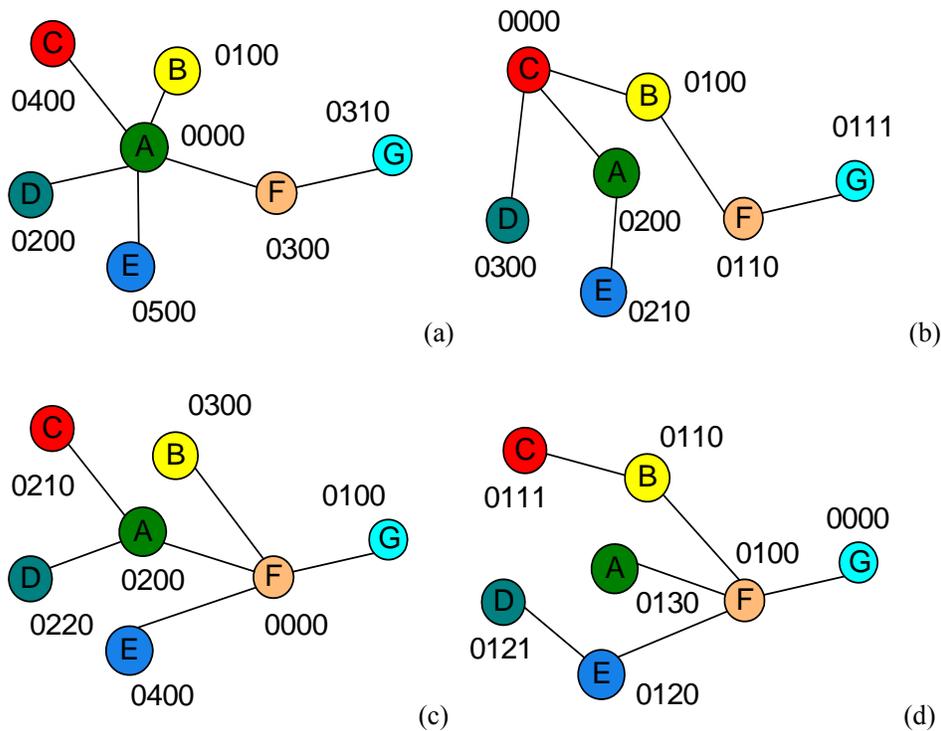


Figure 3.4 Same network with different coordinators

Figure 3.4 illustrates how the Mesh topology in Figure 3.2 is converted into a Tree topology, but each time the coordinator is a different device. In Figure 3.4a and Figure 3.4c the furthest child has only

one parent between them and the coordinator, but in Figure 3.4b and Figure 3.4d there are some branches with two parents. From this it becomes necessary that the protocol must have the capability to force a coordinator. This will eliminate the possibility, if devices are not mobile, that the network may become unnecessarily complex.

Devices will need static addresses to reference the short address otherwise it will be impossible to locate a device if its short address is constantly changing. The protocol will use the same standard as Ethernet by using MAC addresses. After the network is discovered, the network needs to be monitored by the parents to make sure that the network has not changed.

Above solution will resolve the following requirements:

- Discover a manageable logical topology from a Mesh topology.
- Devices must be able to discover the network automatically, move around and recover from power failures.
- No single point of failure.

### **3.1.3 Send packets to correct destination**

In a Tree topology routing is much easier than in a Mesh topology. The protocol must just determine on which branch of the tree the destination resides. Figure 3.5 illustrates the paths that the packet could take to reach two different destinations. When a device receives a packet it must only calculate whether the packet must be sent to a specific child or to its parent.

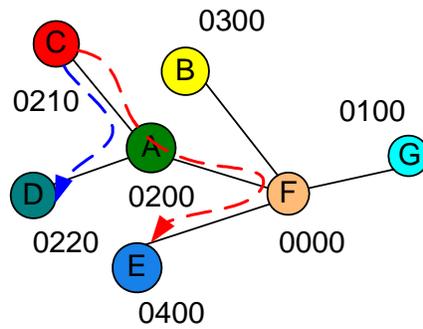


Figure 3.5 Logical Tree topology routing

Because of the Hierarchical topology no routing tables are needed and routing can be done based on short addresses. All devices in the network can act as routers.

### 3.1.4 Devices control each other

The protocol needs to be able to setup links between device inputs and outputs. Before a link can be established there must be a verification process to determine if the two links can control each other. An example is that a light switch cannot activate an alarm system.

## 3.2 Commands

### 3.2.1 Overview

The protocol can be broken into three main components:

- Device discovery and maintenance

At this layer the protocol must be able to elect a coordinator, add new devices in a Tree topology and associate the MAC address with a short address. Reachability between parent and children must be verified continuously and inactive relationships must be cleaned up.

Routing is not necessary because devices only communicate with its neighbour devices.

- Establishing connections and maintenance

All new connections in the process to be established between two control points must be recorded with the coordinator. The coordinator must be able to administer multiple non-established connections in the process. The coordinator will verify that only similar control points can establish a connection between themselves. When a connection is established, the coordinator must handover control of the connection to the hosting devices of the control points. From there it is the responsibility of the hosting devices to update each other about their status. When an input changes, the local device will send the new value to the destination device and if no change occurs in a pre-defined time the devices must update each other. If the short address of a device changes, then they must be able to locate the device and record the new short address. Established links must be saved so that the network can recover from a power failure.

- Security and reliability

Data can be changed in a wireless network accidentally or deliberately. Data is changed accidentally when the transmission medium is faulty and Cyclic Redundancy Check (CRC) is used to identify corrupted data packets. Hackers change data deliberately to access information or to corrupt a system and encryption is used to protect data against this type of attacks.

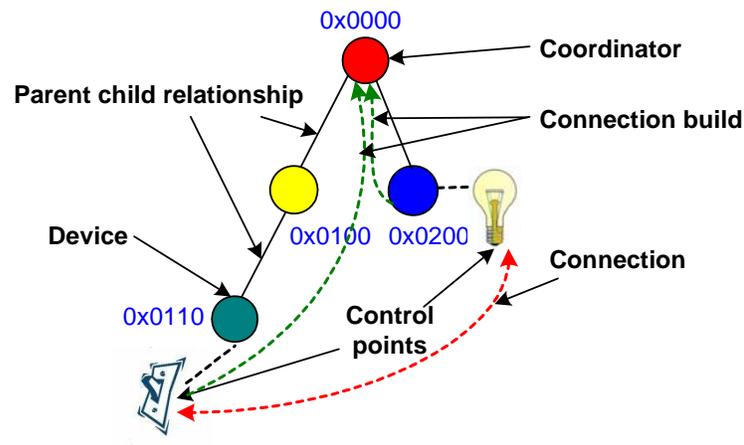


Figure 3.6 Overview of connections

Figure 3.6 illustrates the difference between a control point and a device and between a parent-child relationship and a connection.

### 3.2.2 Device discovery and maintenance

#### 3.2.2.1 Discover short address

The Request short address is responsible to convert the Mesh topology into a Tree topology. The commands will be responsible for the following:

- Request short address;
- Offer short address;
- Accept the best offer;
- Protecting the coordinator's short address;
- Manage changes in device reachability.

The simplest way to explain the Request short address command will be to investigate various scenarios.

### Scenario 1

The network consists of two devices that are in range of each other.

#### Solution:

Figure 3.7 illustrates two devices that use Request short address commands to establish a network. When Device 1 is switched on, it requests a short address. If it does not get any offers after a pre-defined timeout, it elects itself as the coordinator. When Device 2 is switched on, it will request a short address and Device 1 offers address 0x0100. After a pre-defined timeout, the requester accepts the best address. In this case the only offer was 0x0100.

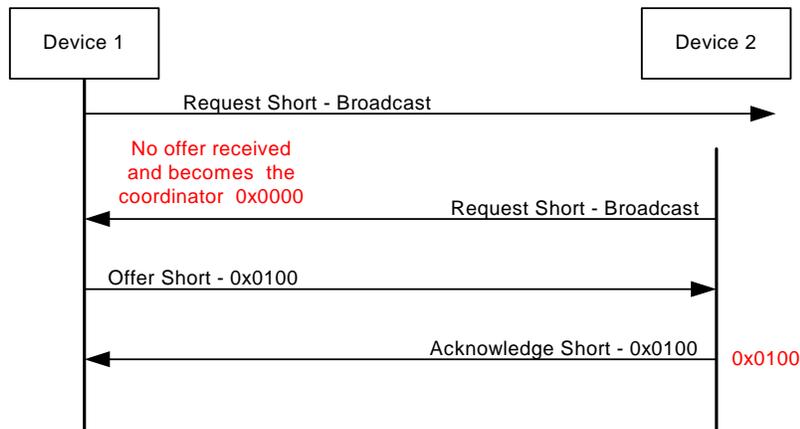


Figure 3.7 Request short address with two devices

After a power failure all devices will start at the same time. A random timer allows devices to request short addresses gradually after its power is connected or restored.

### Scenario 2

The network consists of three devices that are in range of each other. Device 1 is powered on first then Device 2 and Device 3 last.

*Solution:*

Figure 3.8 illustrates the three devices using Request short address commands, to establish a network. Until Device 2 accepts offer 0x0100, it is the same as in scenario one. When Device 3 requests a short address, it first receives an offer 0x0110 from Device 2. Device 2 will wait for a pre-defined period for a better offer before it will accept 0x0110. In this period it receives an offer 0x0200 from Device 1 and drops offer 0x0110. After the waiting period expires, Device 3 acknowledges 0x0200 with Device 1. Before Device 2 can offer another address the first offer must expire. If a device sends an offer, it must wait for a pre-defined period before the offer expires.

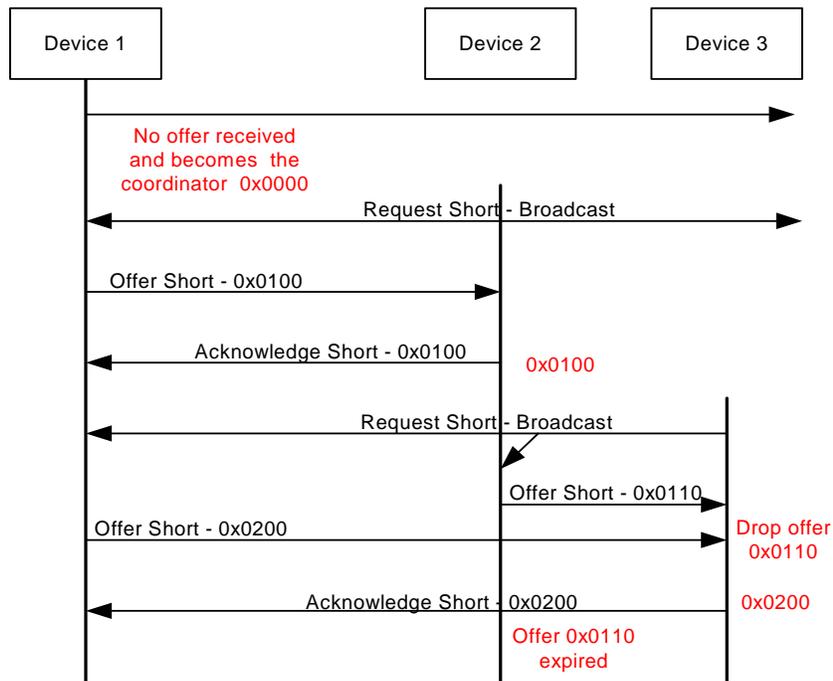


Figure 3.8 Request short address with three devices

**Scenario 3**

The network consists of three devices; only Device 2 is in range of Device 1 and Device 3. Device 1 is powered on first and Device 3 last.

*Solution:*

The solution is the same as with Scenario 2 until Device 2 offers address 0x0110. Device 1 will not offer address 0x0200 because it did not receive an offer. After a pre-defined period Device 3 will accept offer 0x0110.

**Scenario 4**

The same as in Scenario 3, except that Device 1 is powered on first, then Device 3 and last Device 2.

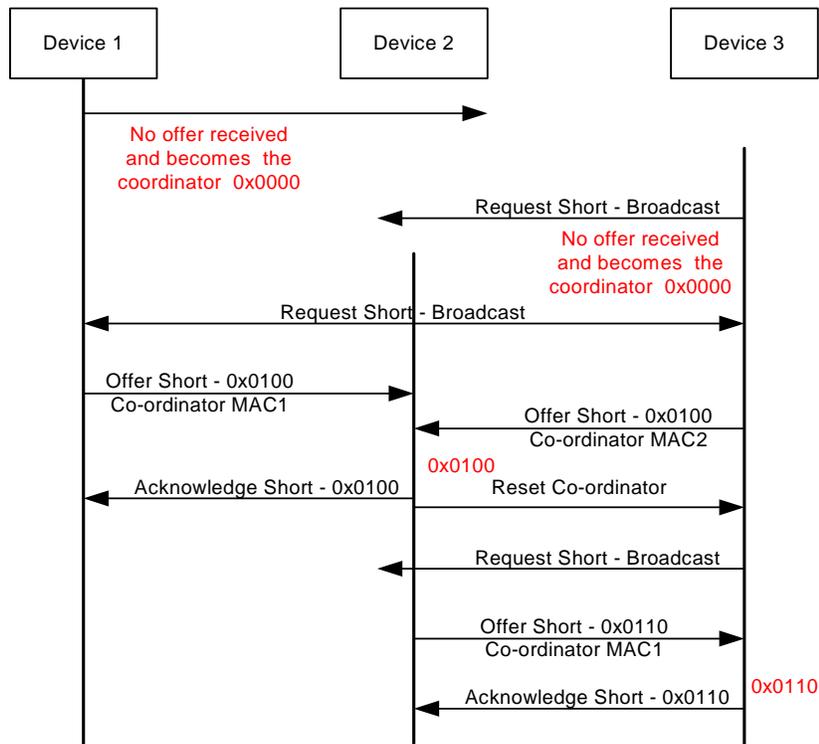


Figure 3.9 One-device join two Personal Area Networks together

*Solution:*

The solution needs to identify the coordinator in the Request short address command. Figure 3.9 illustrates when a device joins two coordinators together. Devices 1 and Device 3 are powered on, but are not in each other's range. Both become coordinators for devices in their range. When Device 2 is

power on it joins the two PAN's together. Device 2 receives an offer 0x0100 from both coordinators but in the offer the coordinator's MAC addresses are specified. Device 2 first acknowledges the previous best offer and sends out a Double coordinator command to reset the short address of the second offer. Device 3 requests a short address after Device 2 had accepted address 0x0100. Device 2 offers address 0x0110 and Device 3 accepts it after a pre-defined time.

### **Scenario 5**

Same as in Scenario 2 except that Device 3 is configured to always become the coordinator when it is powered on.

Solution:

In Figure 3.10 Device 1 is powered on and did not receive any offer after it requested a short address. It elects itself as coordinator. When Device 2 is powered on it only receives one offer, 0x0100 from Device 1, after requesting a short address. When Device 3 is powered on, it elects itself as coordinator and broadcasts a Reset short address command to all neighbouring devices. Device 1 and Device 2 wait for a random time and in the above example Device 2 requests a short address before Device 1. The process will now be the same as in Scenario 2.

If Device 1 requested a short address before Device 2 and it was not in range of Device 3 the same would occur as in Scenario 4.

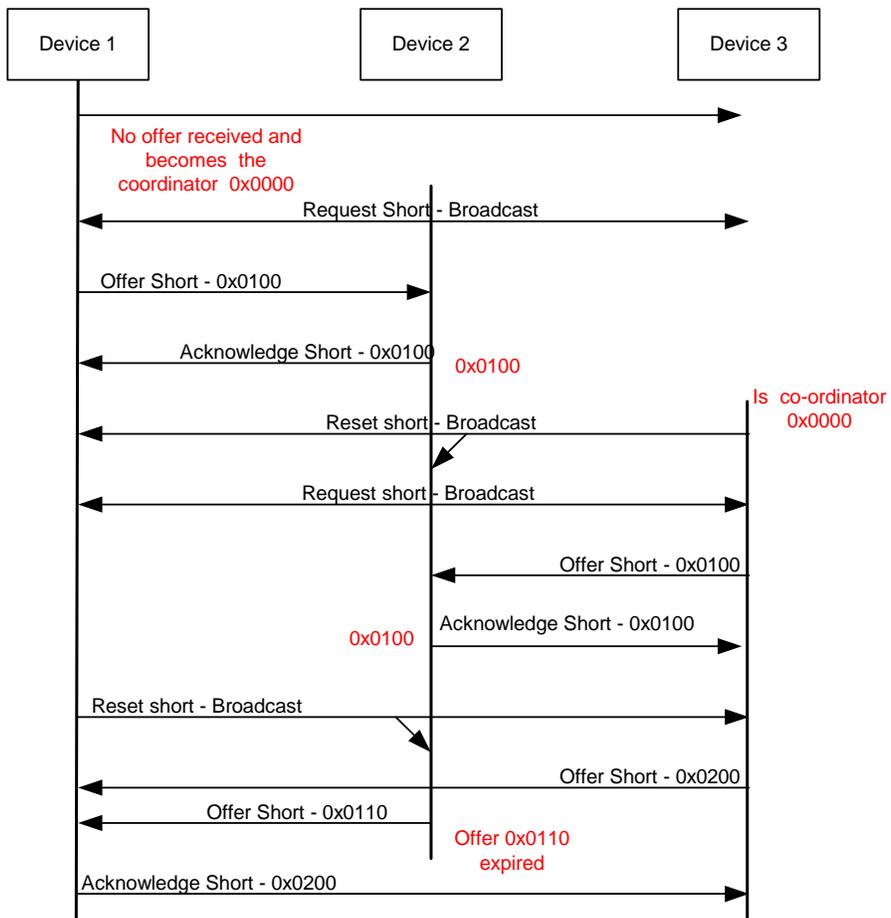


Figure 3.10 Coordinator pre-defined

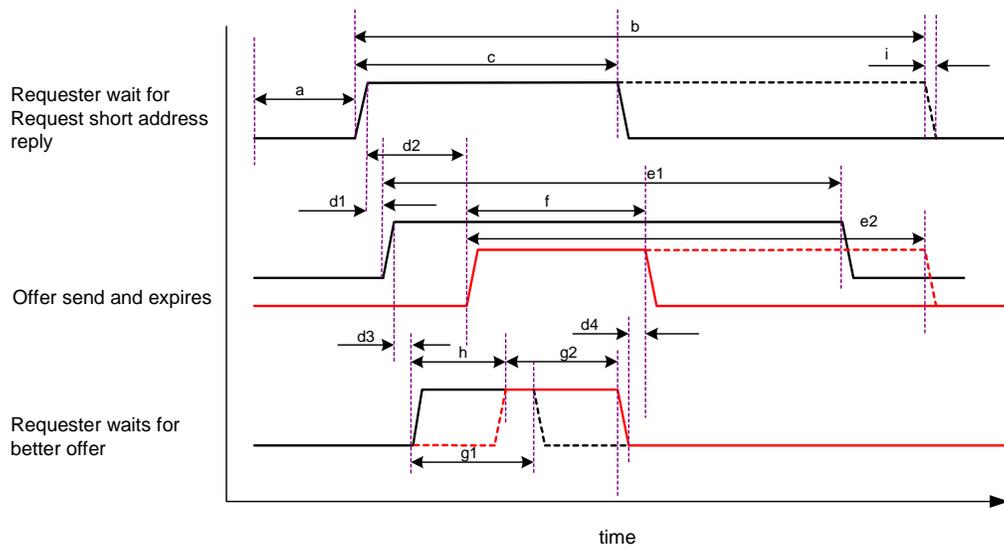


Figure 3.11 Timing schedule for Request short address procedure

Figure 3.11 illustrates the timing relationship of a device that receives two offers from a Request short address. Breakdowns of all the intervals are given below:

- a: Random delay after power was restored or after short address was reset.
- b: Maximum time to wait for offer to be acknowledged. If the time has expired, the device becomes the coordinator.
- c: Time b does not expire and is reset when a short address is accepted.
- d(x): Transmission delay of radio's. The time will decrease when the transmission speed increases.
- d1: Delay from Request short address is transmitted until remote Device 1 receives request.
- d2: Delay from Request short address is transmitted until remote Device 2 receives request.
- d3: Delay from Device 1 transmits an offer until requester receives offer.
- d4: Delay from when requester accepts offer until Device 2 receives acknowledgement.
- e1: Maximum time Device 1 will wait for requester to accept offer.
- e2: Maximum time Device 2 will wait for requester to accept offer.

- f: Offer was accepted.
- g(x): Requester will wait for a better offer.
- g1: Time wait after first offer.
- g2: Time wait after second offer.
- h: Better offer was received, restart timer.
- i: Micro-processor delays.

### **3.2.2.2 Keepalive**

The Request short address command will build the topology, but it will not maintain the topology. When devices are mobile the parents and children can change continuously and the protocol needs a mechanism to keep the Tree topology updated. The Keepalive command will allow parent and child devices to poll each other. The command will be responsible for:

- Maintaining current child and parent table on devices.
- Remove inactive devices from neighbour tables.
- Investigate for better short addresses.

When two devices have a parent-child relationship they will poll each other the entire time to ensure that they are still in range. If the parent poles the child in a pre-defined time the child must acknowledge the request. Both devices must reset their Keepalive timer. When one of the devices is powered off and the other device and does not get an acknowledgment back it must fast poll the other device three times and if no reply was received in that time, then the entry must be deleted from the table as illustrated in Figure 3.12.

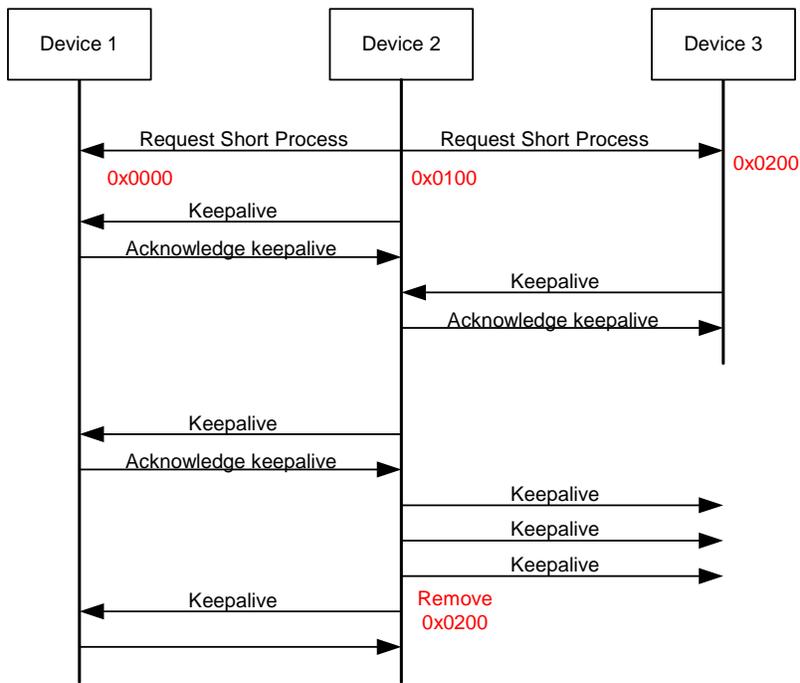


Figure 3.12 Keepalive illustration

Keepalive and Request short address commands can be used together to maintain the topology. In Figure 3.13 Device A is the coordinator, Device B 0x0100, Device C 0x0200 and Device D 0x0110. Device B and C could each have address 0x0100 depending on which device requested the first short address. If Device B lost connectivity and Device A sends a Keepalive, it will not receive any feedback from Device B and remove entry 0x0100 from its neighbour table. Device C requests a better short address after a pre-defined time. The coordinator removes short address 0x0200 and offers short address 0x0100 to Device C. Device C accepts the address and clears its entire neighbour address table. Device D's Keepalive to Device B would have reached a timeout by now and because it has lost its parent it will request a new short address. Device C now offers the same address 0x0110 to Device D.

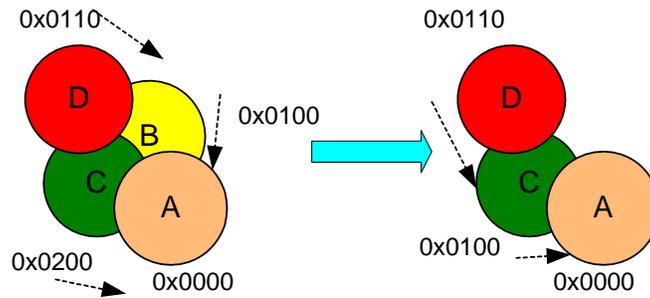


Figure 3.13 Request short address and Keepalive commands builds topology example 1

If Device D's Keepalive request timed out before Device C requested a better short address, the end result would be the same, but the flow of events are different. Firstly Device D would get address 0x0210 from Device C. After a pre-defined timeout Device C requests a better short address and accepts the new 0x0100 address. Device D's Keepalive to Device C's old short address 0x0200 times out and it requests a new short address because it lost its parent. It will accept Device C's offer 0x0110, as illustrated in Figure 3.14.

This technique allows devices to be mobile because if a device moves out of one device's reception area into another reception area, it can request a new short address. The maximum speed that a device can move will be determined by how fast the devices can poll each other. This again is determined by the transmission speed of the radio modem and the amount of devices that have joined a network.

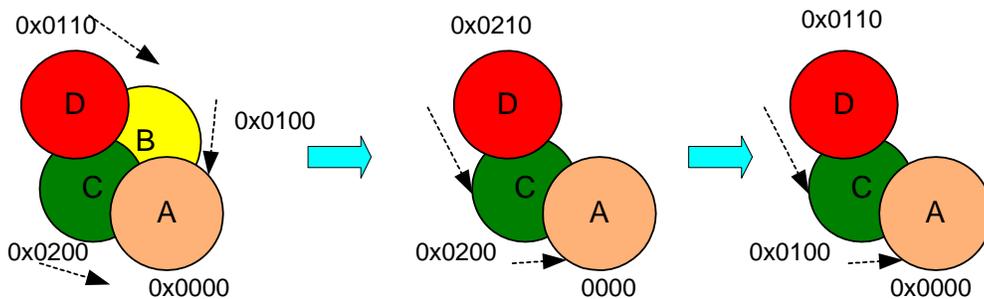


Figure 3.14 Request short address and Keepalive commands builds topology example 2

### 3.2.3 Establishing connections and maintenance

#### 3.2.3.1 Forward packets

After the topology is manageable, the protocol can forward packets between end devices. Devices can make one of three routing decisions as illustrated in Figure 3.15 when a packet flows from Device C to E:

- Route to parent – Device A & C.  
     Route to specific child – Device F.
- At destination – Device E.

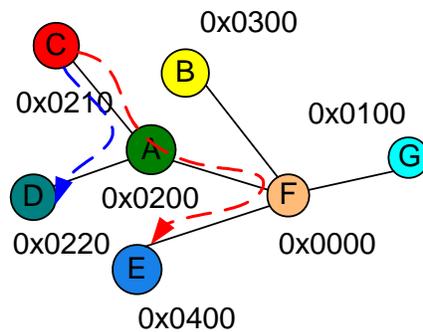


Figure 3.15 Routing decisions

The protocol must make provision for specifying the short addresses of the source, forward source, forward destination and destination. If the short address fields are investigated for each device in the path between Devices C and E the following short address values can be assumed:

Table 3.1 Short addresses used to forward packet for logical tree

Device	Source	Forward Source	Forward Destination	Destination
C	0x0210	0x0210	0x0200	0x0400
A	0x0210	0x0200	0x0000	0x0400
F	0x0210	0x0000	0x0400	0x0400
E	At Destination			

When a protocol allows packets to be forwarded, a mechanism must be established to discard a packet that cannot reach its destination. Internet Protocol (IP) uses a technique called Time To Live (TTL) [14], with a maximum value of 255. Every router that forwards this packet will increment the TTL value with one.

This protocol will use the same technique but the maximum initial value can be calculated as follows:

- Maximum hops from coordinator to device  $0x0EEE = 3$ .
- Maximum hops from end device to end device  $= 3 \times 2 = 6$ .

If two PAN's are allowed to communicate, the TTL must be increased with one to seven. (This is out of the scope of this research.) A device will discard a packet with a TTL of zero if it has not yet reached its destination.

### **3.2.3.2 Establish connections between input and output control points**

The coordinator must manage all new connections established between two control points. The coordinator must be able to verify that only similar control points can establish a connection between themselves. To do this the protocol must be able to identify the control points to the coordinator.

The following characteristics of the control point must be identified:

- *Device short address*

The coordinator must exchange the two short addresses to the end devices that have established a connection.

- *Device MAC address*

If a device's short address has changed while it was busy establishing a connection, then the other device can discover the device's new short address via its MAC address after the coordinator have exchanged information.

- *Identification number*

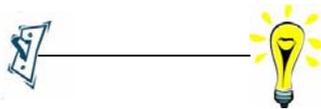
Each device can host multiple control points. Each control point must have a unique Identification (ID) number. This makes it possible to create a connection between, Device A's control point six and Device C's control point two.

- *Group*

Each control point must belong to a group. Groups can include security, lighting or irrigation. Groups must be able to grow as required. Only control points belonging to the same group can establish a connection between each other, e.g.: A light switch must not be able to control the irrigation system.

- *Input/Output type*

An input control point can only control an output control point. Some outputs can only accept one input like a room light switch and light as illustrated in Figure 3.16.



*Figure 3.16 Light switch*

Other outputs can be controlled via two inputs such as a passage light with two switches. In this case either switch can switch the light on as illustrated in Figure 3.17.

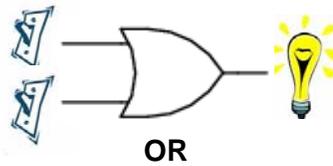


Figure 3.17 Passage light switch

To prevent wasting water an irrigation system must only start irrigating when it is not raining as illustrated in Figure 3.18.

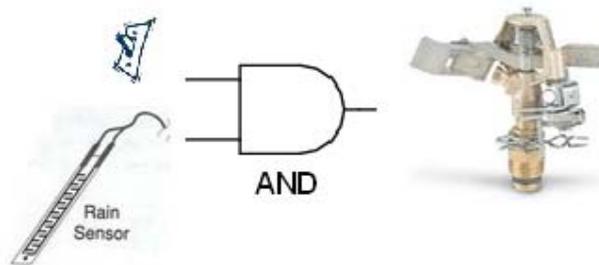


Figure 3.18 Irrigation system with rain sensor

- Capability

Even if both control points belong to the same group and one is an input and the other an output, they must support the values that are sent to them. For example: A rain meter sensor is connected as an input. Some rain sensors can only indicate if it is raining or not, while others can indicate how much it has rained. Some input and outputs only have an on/off condition, while others have a range of values.

- Key

The coordinator must be able to administer multiple non-established connections in progress. To make this possible the coordinator must be able to identify what control point needs to establish a connection with whom. When a control point registers with the coordinator it will happen by means of a key. The coordinator will only attempt to establish a connection

between two control points with identical keys. The key is only valid while the connections are established.

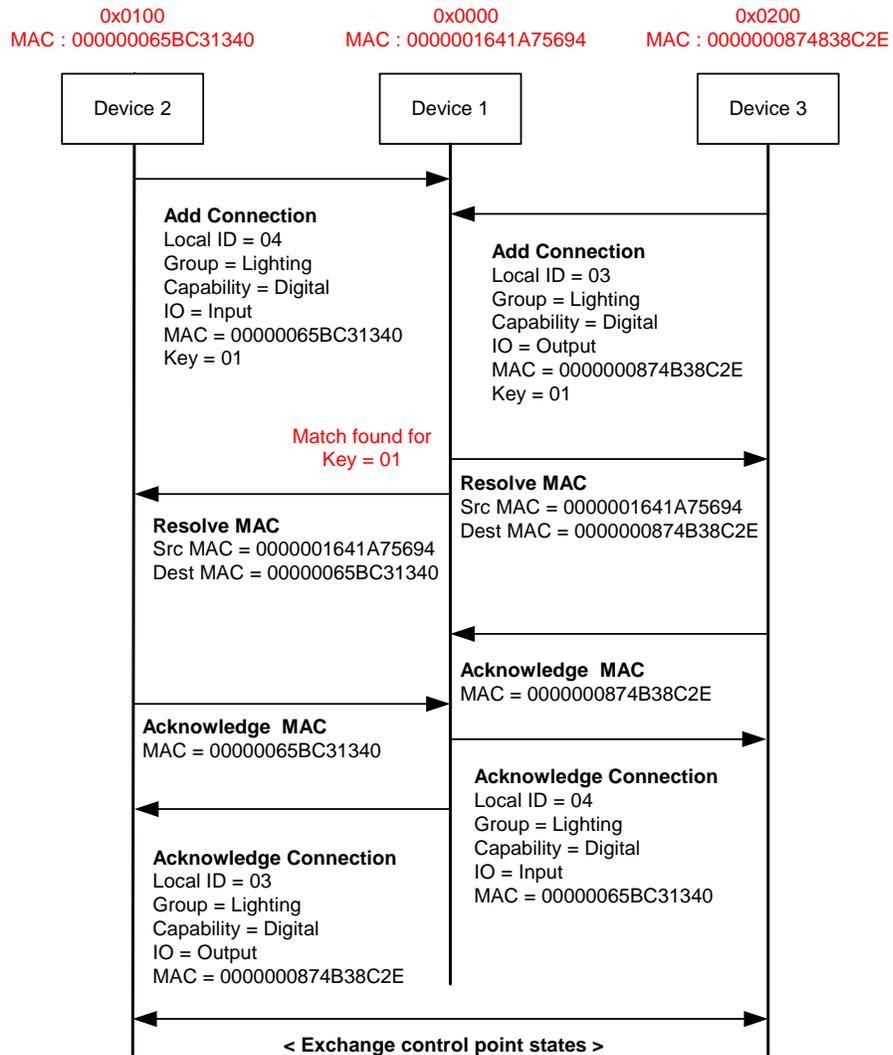


Figure 3.19 Establish connection successfully

When a connection is established, the coordinator will first confirm that both devices' short addresses have not changed, before it hands over control of the connection to the hosting devices of the control points. Figure 3.19 illustrates the process when a light switch on Device 1 establishes a connection to a light on Device 3. There is still the possibility that the device will request a better short address after

an other device has done a MAC Resolve command as described in the section about Keepalives in section 3.2.2.2. A mechanism must be built into the protocol to disable this function temporarily if the device is busy sending data.

If the coordinator does not find a partner for a request in a pre-defined time, it must send a message to the device informing it that the request was unsuccessful. The process will be the same as in Figure 3.19 except that the coordinator will not send a Resolve MAC address to the hosting devices, but a Negative Acknowledgement.

If a connection needs to be terminated, the two hosting devices will communicate the termination between themselves without interacting with the coordinators. Packets will always be routed via the parent or child as described in section 3.2.3.1. For devices with a short address with format 0x0X00 the coordinator is also their master. Before the connection is terminated the device that wants to terminate the connection needs to confirm the remote devices short addresses by using the MAC Resolve command as illustrated in Figure 2.20.

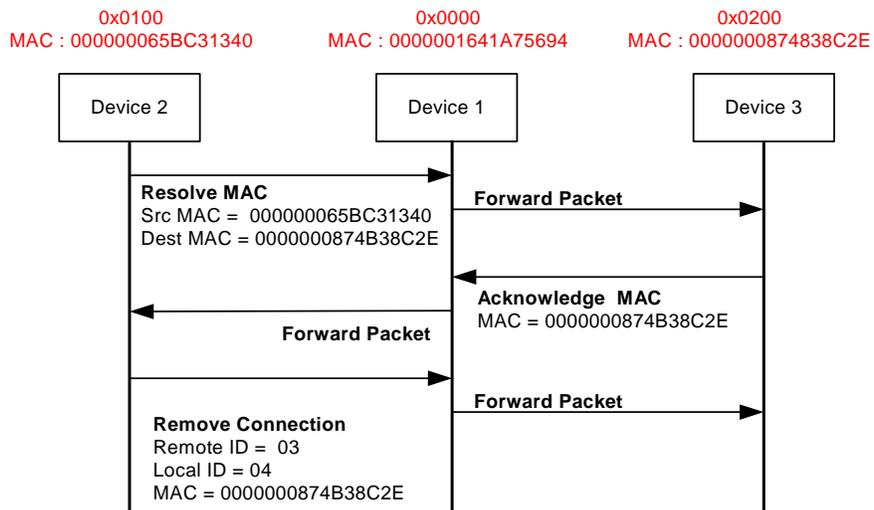


Figure 3.20 Terminate connection

All connections established must be stored locally on the hosting device.

### **3.2.3.3 Update status and Keepalives**

Both control points' hosting devices need to inform each other's about its state and description. When a control point condition changes it will inform the remote device. When the output control point type is not 'AND' or 'OR', then only the input control point state can change. But when the output type is 'AND' or 'OR' then the output control point needs to inform the second passive input control point that its condition has changed.

A status update will be sent to the remote hosting device, if the state has not changed for a pre-defined period. This will ensure that the connections between the control points are active because the remote device must reply back with its control point state. When the local hosting device does not receive a reply, it will try to discover the remote device's new short address. The local device will broadcast out a MAC Resolve packet to all neighbour devices, requesting the MAC address owner to reply with its short address. If the device receives a new address it will update its local information. The device must disable and not remove the remote value if no reply is received. The local device will continuously try to re-establish the connection as illustrated in Figure 3.21.

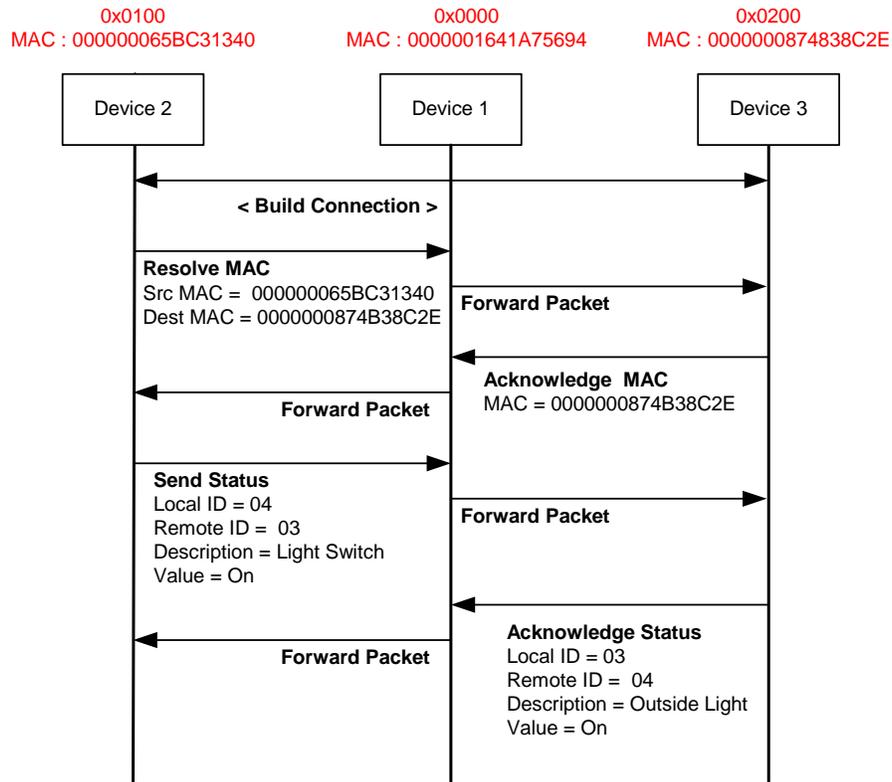


Figure 3.21 Send status

When a device's short address has changed, it is the Resolve MAC command's responsibility to discover the new short address as illustrated in Figure 3.22.

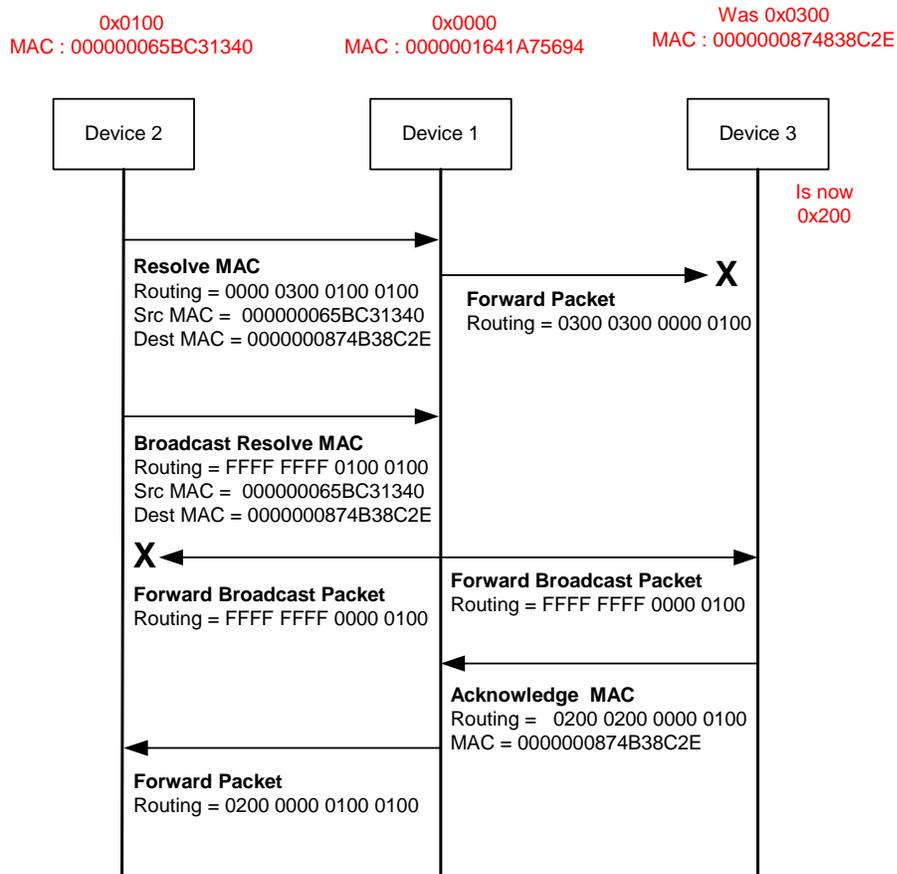


Figure 3.22 MAC Broadcast rediscovers changed short addresses

The device will first send a MAC Resolve directly to the remote device. If no reply is received in a pre-defined time, the local device will broadcast the MAC Resolve command to all devices. These devices will again forward the broadcast to their neighbour devices. Care must be taken to prevent a device resending a broadcast.

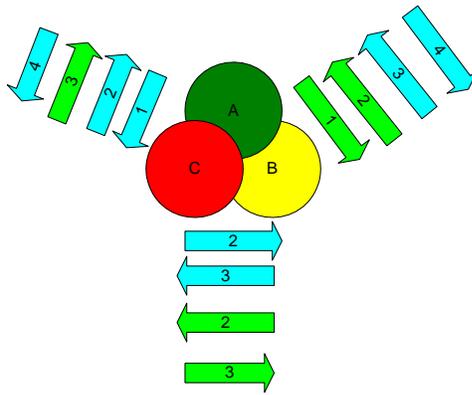


Figure 3.23 Broadcast storm

If there are no mechanisms to prevent resending of broadcast packets, the packet will just be sent in a loop until the Time to Live value is zero as illustrated in Figure 3.23. The problem can be resolved if a device does not accept any broadcasts for a pre-defined time after it had sent a broadcast as illustrated in Figure 3.24.

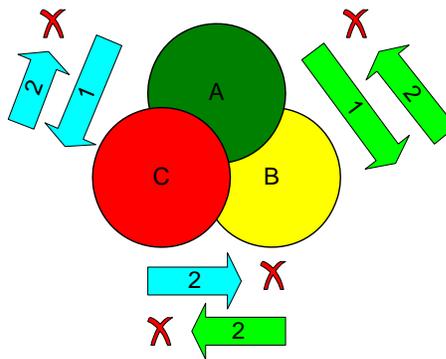


Figure 3.24 Broadcast avoidance

### 3.3 Summary

The protocol's main functions are to convert the wireless Mesh topology into a controllable Tree topology. It must use a hierarchical short address design to eliminate the use of a complex routing process. Only one device, namely the coordinator, can control the environment. When contact to the

coordinator is lost, any other device can take over the functions of the coordinator. Keepalives make it possible for devices to sense topology changes.

A device can have more than one control point. Only similar control points can create connections between themselves. The Resolve MAC address command makes it possible for the upper layer to track short address changes in the lower layer.

# Chapter 4

## 4 Device discovery and maintenance

In Chapter 4 and 5, the requirements discussed in Chapter 3, will be used to develop the frame format and software of the new synchronisation of data method. The Device Discovery suite is an interface between the hardware and the Establish Connection suite.

### 4.1 Simulate wireless

No test equipment is available when a new protocol is developed. It is up to the developer to develop test equipment or software to evaluate and to generate results for the new protocol. It creates even more questions to test such protocol wireless. For this reason, the protocol was first developed for a personal computer (PC). To simulate wireless transmission, packets will be encapsulated in a UDP/IP (User Datagram Protocol/Internet Protocol) broadcast packet. UDP is an unreliable protocol and broadcasts will send data to all computers on the same IP network. UDP will require the upper layers to accept the correct packet and ensure that the data is not corrupt. The wireless medium will require the same functionality from its upper layers. The computer software was developed with Microsoft Visual Basic 2005 Express edition.

### 4.2 Layout of program

The program is divided into four main tabs:

- Endpoint Setup
- Global Setup
- Trace
- Simulate

## 4.2.1 Endpoint setup

Define Endpoints

	Description	Group	Capability	I/O Type
1	Door Sensor	1,Security	1,Digital	1,Input
2	Siren	1,Security	1,Digital	3,Output (And)
3	Light Switch	3,Lighting	1,Digital	1,Input
4	Outside Light	3,Lighting	1,Digital	2,Output (Single)
5	Grass Moisture Front Sensor	4,Irrigation	1,Digital	1,Input
6	Siren	1,Security	1,Digital	4,Output (Or)
7	Window Sensor 1	1,Security	1,Digital	1,Input
8	Irrigation Valve 1	4,Irrigation	1,Digital	3,Output (And)
9	Temperature Sensor 1	1,Security	2,Analogue	1,Input
10	Dimmer Switch	3,Lighting	2,Analogue	1,Input

Figure 4.1 Define the endpoints

Each device can host multiple endpoints and the program was written to simulate ten endpoints. The description, group, capability and input/output type can be defined for each endpoint. Under the group selection security, air-conditioning, lighting and irrigation are choices that can be selected. Under the capability tag digital or analogue can be selected and under the I/O type it is input, output (single), output (and) and output (or) as shown in Figure 4.1. When a device is built in hardware these values will be able to change, but it will depend on the hardware, e.g.: If a device is built to be a light switch, it will only have one input endpoint that belongs to the lighting group. Groups, capabilities and I/O type can be added as long as there is a standard between all devices. This section will be described in detail in Chapter 5.

## 4.2.2 Global setup

The screenshot shows a configuration window for a network device. It is split into two main areas. The left area, labeled 'Settings', has three sub-sections: 'Duplex' with radio buttons for 'Full Duplex' and 'Half Duplex'; 'Coordinator' with three checked checkboxes: 'Double Coordinator Protection', 'New Coordinator Protection', and 'Force Coordinator'; and 'Allow Short Address' with four checked checkboxes: 'All', '0x00', '0yx0', and '0yxx', and an unchecked checkbox for 'Lowest Address is 05xx'. The right area, labeled 'IP', has a checked 'Enable' checkbox, followed by input fields for 'Server IP Address' (10.43.221.22), 'Server MAC Address' (0000001641A75694), and 'Server Virtual MAC Address' (3964911644015710). There is also a checked 'Use Virtual MAC' checkbox and a 'UDP Port (Default 5001)' field set to 5700. Below the IP section is a 'Serial Port (8 N 1)' section with a checked 'Enable' checkbox, a 'Port' dropdown menu, a 'Serial Virtual MAC' field (3783319240937501), and 'Connect' and 'Disconnect' buttons.

Figure 4.2 Global setup

The Global setup tab consists of three sections as shown in Figure 4.2.

In the settings section full and half duplex can be set to simulate different types of radio modems. When half duplex is selected the receiver path is disabled while data is transmitted.

The coordinator protection and Force coordinator functionality are discussed in detail in section 4.5.1.

The Allow short address tab is used to simulate the range of the radio modem because on an Ethernet cable all UDP broadcasts will be forwarded to all devices in that IP network. If you only enable the 0x0yx0 tab, the computer simulation will ignore all short addresses from the coordinator and only accept short addresses from devices with an 0x0[1-E][1-E]0 address. The Lowest address is 0x05xx tab that is used to simulate a device requesting a better address. When the tab is selected the parent will offer 0x0500 and when it is deselected the device will offer 0x0100. This is explained in section 4.5.2.3 under the Request short address command.



## 4.2.4 Simulate

The screenshot displays the 'Simulate' tab interface, divided into three main sections: 'Local', 'Remote', and 'Link'.

- Local:** A table listing various control points with their status values. Each entry includes a radio button for linking and a text input for the status value.
- Remote:** A table showing a single remote control point with its address, description, and a visual status indicator (a bar of green dots).
- Link:** A control panel for managing connections, including a 'Key' input field, 'Link', 'Delete', 'Reset', and 'Change Value' buttons.
- Connections In Progress:** A list of connections currently being established, all labeled as 'Spare'.

Local		
Description	Link	Status
Door Sensor	<input type="radio"/>	255
Siren	<input type="radio"/>	
Light Switch	<input type="radio"/>	255
Outside Light	<input type="radio"/>	
Grass Moisture Front Sensor	<input type="radio"/>	255
Siren	<input type="radio"/>	
Window Sensor 1	<input type="radio"/>	0
Irrigation Valve 1	<input type="radio"/>	
Temperature Sensor 1	<input type="radio"/>	0
Dimmer Switch	<input type="radio"/>	50

Remote		
Remote Address	Description	Status
000000874B38C2E/02/0100/65	Siren	<div style="width: 100%; height: 10px; background-color: green;"></div>

Link

Key:

Link Delete

Reset

Change Value

Connections In Progress

- Spare

Figure 4.4 Simulate

Figure 4.4 shows the Simulate tab where the device's endpoints are configured. From this tab connections can be established to any other control points in the PAN. Unique keys can be entered, the status of the input control point can be changed and all local and remote control point's values are displayed. If the device is the coordinator, all the connections that are in the process of being established will be displayed.

## 4.3 Cycle redundancy check

Cycle redundancy check will be used in all data frames transmitted by the devices.

The mathematical calculation is the division of data by the polynomial. Hardware division uses modulo-2 arithmetic and is performed by an exclusive-OR gate and serial shift registers. The XOR function output will be a '1' when the inputs differ as illustrated in Figure 4.6.



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

Figure 4.5 XOR Gate

The first sixteen data bits are first loaded into the serial shift register as illustrated in Figure 4.6. If  $x^{16}$  is a '1' the register is XOR with the polynomial and then shifted left, but if it is a '0' the register is only shifted left [8].

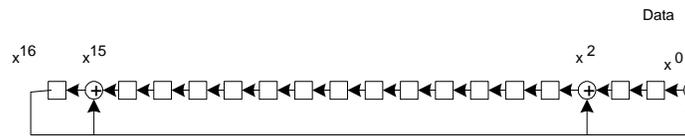


Figure 4.6 CRC-16 generator

#### 4.4 Control frames

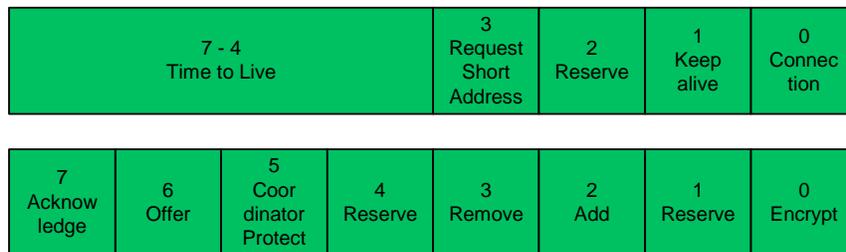


Figure 4.7 Control frame

Figure 4.7 illustrates the control frame that will be used by the protocol. Each field will be discussed in detail in the next two chapters.

Each time a device forwards a packet, it will subtract one from the Time to Live (TTL) value. When the value is zero the device will discard the packet if it has not reached its final destination. The functionality of TTL, except for the value, is the same for IP [14]. This was explained in detail in section 3.2.3.1.

## 4.5 Discover short address function development

The short address function enables the restructuring of all devices into a manageable Tree topology.

### 4.5.1 Frame format

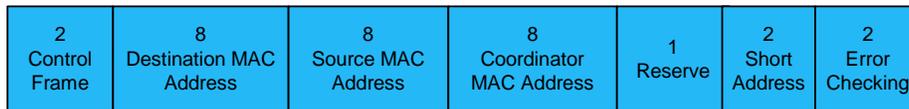


Figure 4.8 Discover short address frame

Figure 4.8 illustrates the Discover short address frame. The number above each block indicates the length in bytes of each field. Bit 3 of the first control field identifies the Discover short address frame as illustrated in Figure 4.7. The destination, source and coordinator MAC address must always be identified. The value of any unknown field will be 0xFFFFFFFFFFFFFFFF for MAC addresses and 0xFFFF for the short address. The error-checking field uses CRC-16-CCITT. The Discover short address suite consists of five commands. The functionality of the commands was discussed in section 3.2.2.1:

- Request short address

8800	FFFF FFFF FFFF FFFF	84F2 6DF4 3749 04A1	FFFF FFFF FFFF FFFF	AA	FFFF	CRC
------	------------------------	------------------------	------------------------	----	------	-----

Figure 4.9 Request short address frame sample

- Offer short address

8840	84F2 6DF4 3749 04A1	84F2 8672 4D7B 0B65	84F2 6C3E 3E57 1536	AA	0110	CRC
------	------------------------	------------------------	------------------------	----	------	-----

Figure 4.10 Offer short address frame sample

- Acknowledge short address

8880	84F2 8672 4D7B 0B65	84F2 6DF4 3749 04A1	84F2 6C3E 3E57 1536	AA	0110	CRC
------	------------------------	------------------------	------------------------	----	------	-----

Figure 4.11 Acknowledge short address frame sample

The first three commands always function together. With reference to the above frames, a broadcast is when all the values in a field are 0xF. When the device with MAC 0x84F26DF4374904A1 broadcasts a Request short address command, it receives an offer 0x0110 from the device with MAC 0x84F286724D7B0B65. The requesting device discovered that the coordinator's MAC is 0x84F26C3E3E571536 and it will now send an acknowledgment back.

- Reset short address

8820	FFFF FFFF FFFF FFFF	84F2 6DF4 3749 04A1	FFFF FFFF FFFF FFFF	AA	FFFF	CRC
------	------------------------	------------------------	------------------------	----	------	-----

Figure 4.12 Reset short address sample

When a device detects that there is more than one coordinator in the PAN, it must send to a specific device or broadcast to all devices, to reset their short address and to request a new short address.

- Protect coordinator short address

8820	FFFF FFFF FFFF FFFF	84F2 6DF4 3749 04A1	FFFF FFFF FFFF FFFF	AA	0000	CRC
------	------------------------	------------------------	------------------------	----	------	-----

*Figure 4.13 Protect coordinator short address frame sample*

When a new coordinator is chosen the coordinator will send out a Protect coordinator command. This will disable any Reset coordinator commands for a pre-defined time. This will prevent an endless loop where the children detect that there is a new coordinator and sends a reset coordinator command. The connection between the devices will timeout and request a new short address from the coordinator.

## 4.5.2 Algorithms

### 4.5.2.1 Main program

The main program is part of Chapter 4 and 5. It is discussed here because the Request short address command is part of the initialising processes. When a device is powered on, it needs to set itself up, for example as a light switch. This information can be hard coded or loaded from memory if the functions are allowed to change.

Next, the device must initialise its communication. The simulation software must have an IP address, calculate the broadcast IP addresses, setup a serial port for RS232 communication and calculate a virtual MAC address. The virtual MAC address will either be assigned to the RS232 interface or used

to simulate multiple devices. The physical device will only need to initialise communication with the wireless component.

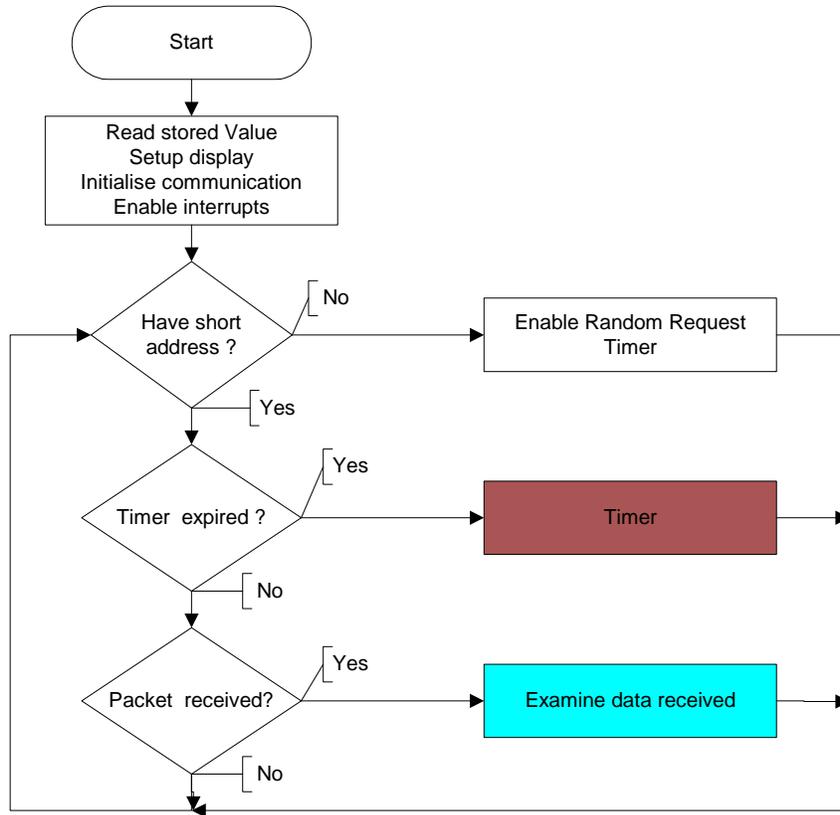


Figure 4.14 Flow diagram of the main program

The last of the initialisation process is to enable the interrupts for the timer expired and data received. Except for the input and output simulation displays, all other functions will be the same for the computer simulator and the physical device.

If a device does not have a short address, a random start timer is enabled. The device will send out a short address request after the timer has expired. The delay before is a mechanism to protect the PAN when multiple devices need to request short addresses simultaneously. The delay will vary between 1 and 1000 milliseconds. This timer will be discussed in detail in section 4.5.2.4.

When the device sends a Request short address, it will start a Wait for offer timer. If this timer expires before an offer was received, the device becomes the coordinator for the PAN. The new coordinator will then send out a Protect coordinator command. All devices will now accept offers from the new elected coordinator.

The software will continuously monitor the timer and data received interrupts. Each of them will be discussed in the following three sections.

#### 4.5.2.2 Data received interrupts

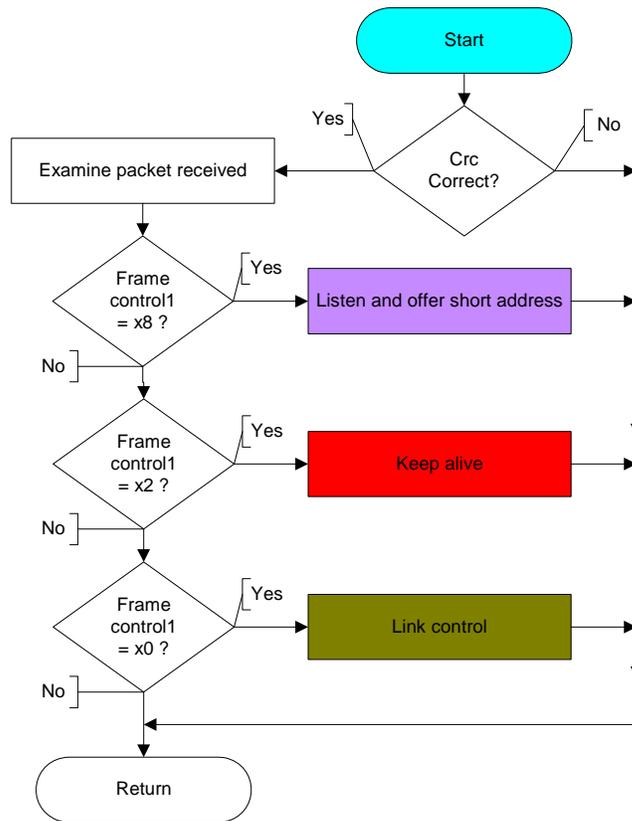


Figure 4.15 Flow diagram for data received interrupt

Data received interrupt is part of the discovery and connection between a control point section.

An interrupt is raised when a data packet is received. The interrupt procedure must first validate with CRC 16, if the packet is not corrupt, then examine control frame one and determine what type of data packet was received. Short addresses will be discussed in the next section, Keepalive in section 4.6 and Link control in Chapter 5.

#### **4.5.2.3 Short address received**

In section 4.5.1 five types of short address frames and functions were discussed. Only the steps of the program for each command will now be discussed and is illustrated in Figure 4.16.

- *Request*

A Request destination MAC address is 0xFFFFFFFFFFFFFFF. The receiving device will discard the packet, if the destination MAC address is not a broadcast or if it has a pending offered short address. The device must also discard a request from its parent because the request was sent by the parent to request a better short address and a child's offer will always be inferior to its the short address the parent device already has. This is also a mechanism to prevent a child device offering a parent a new address after the coordinator was lost and all the devices request new short addresses.

If a device receives a request from a device that is already a child, it will clear the child's existing information. The device can now offer the lowest spare. This offer will expire if the child receives a better offer from another device. With this technique bandwidth is saved; when the device receives a better offer it does not need to send a disconnect command to its current parent.

When the device offers a short address, it will start an Expire offer timer. The other device must accept this short address before the offer expires. This will be discussed later in the section about short address timers in section 4.5.2.4.

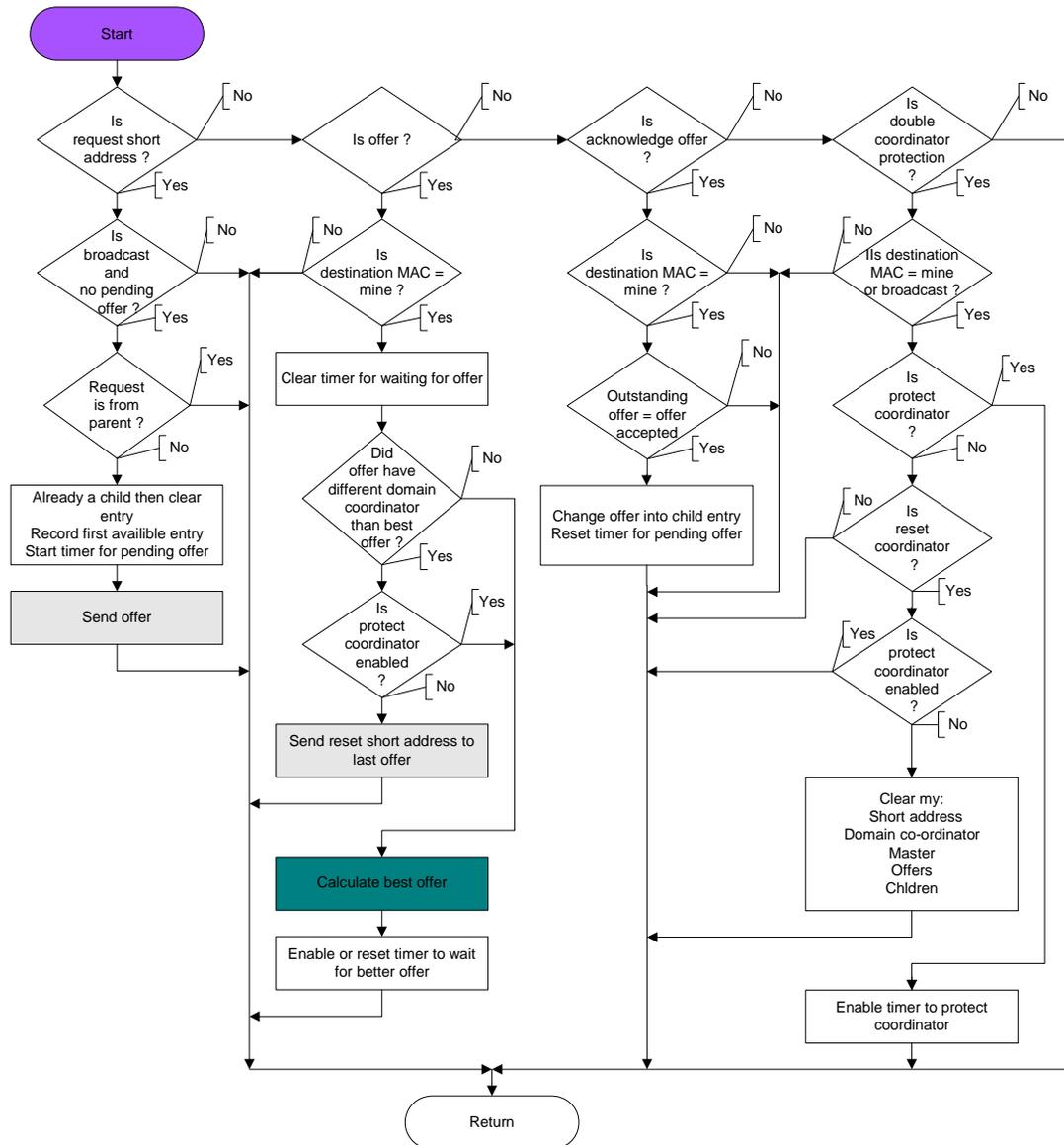


Figure 4.16 Flow diagram for short address data received

- Offer

This packet is the reply to the Request short address that was sent. The offer will only be examined if the packet's destination MAC address is the same as its own MAC address. The device will clear the Request short address timer if it is a valid offer.

The device must now compare the current offer against previous offers. To protect the PAN the requester will verify that the coordinator value in the coordinator field is the same as the previous offers. If it is not the same and the Protect coordinator is not enabled a Reset short address will be sent to the last offered device.

If there is not a problem with the offer's coordinator value, the previous best offer is compared with the new offer and only the best offer is stored. The calculation will be discussed later in this section.

When a new offer is accepted, the Wait for offer timer is enabled, if it is the first offer, or reset if it is a better offer. The above-mentioned timer will be discussed in more detail later in the timer interrupt section. In brief the device will send an acknowledgement to the device with the best short address offer after the timer had expired.

- *Acknowledge offer*

As previously explained, the acknowledgment is the reply to an offer. The acknowledgment will only be examined if the packet's destination MAC address is the same as the device's own MAC address. The device is then recorded as a child and the pending offer timer is reset.

- *Double coordinator protection*

The Double coordinator protection command consists of two parts:

1. Send a Reset short address to a device.
2. Protect a newly selected coordinator.

The command will only be examined if the packet's destination MAC address is the same as the device's MAC address or the destination MAC address is a broadcast. If the command is

Protect coordinator, a timer to protect the coordinator will be enabled, but not if the command is a Reset coordinator. The device will then clear its short address and the main program will request a new short address.

The best short address is the address that is the nearest to the coordinator 0x0000 as illustrated in Figure 3.3. Figure 4.17 illustrates the algorithm for the calculation. Firstly all “0” values before any value bigger than “1” is stripped and the numbers move to the right. The left values are filled with “0” to make up a nibble. It is done for the previous best offer and the newly received offer.

*Table 4.1 Best short address calculation*

Best offer		New offer		New best offer
0x0500	0x0005	0x0140	0x0014	0x0500
0x0121	0x0121	0x05F0	0x005F	0x05F0
0x0300	0x0003	0x0210	0x0021	0x0300
0x0212	0x0212	0x0231	0x0231	0x0212
0x0612	0x0612	0x0900	0x0009	0x0900

The smallest value is the best offer. Figure 4.17 illustrates how the calculation works.

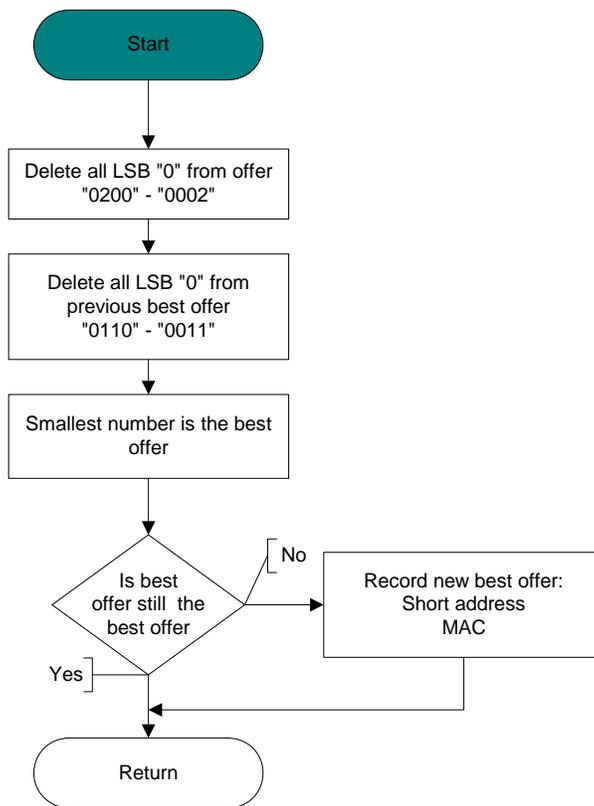


Figure 4.17 Flow diagram to calculate best offer

#### 4.5.2.4 Short address timer

Each suite of commands will have a group of timers, but in this section only the timers associated with short addresses will be explained.

The short address suite has six timers as illustrated in Figure 4.18 and they are:

- *Random start*

When a device is powered on or it must request a short address the Random start timer is enabled. A random value is generated with a maximum of one second and when this timer expires a Request short address command can be sent.

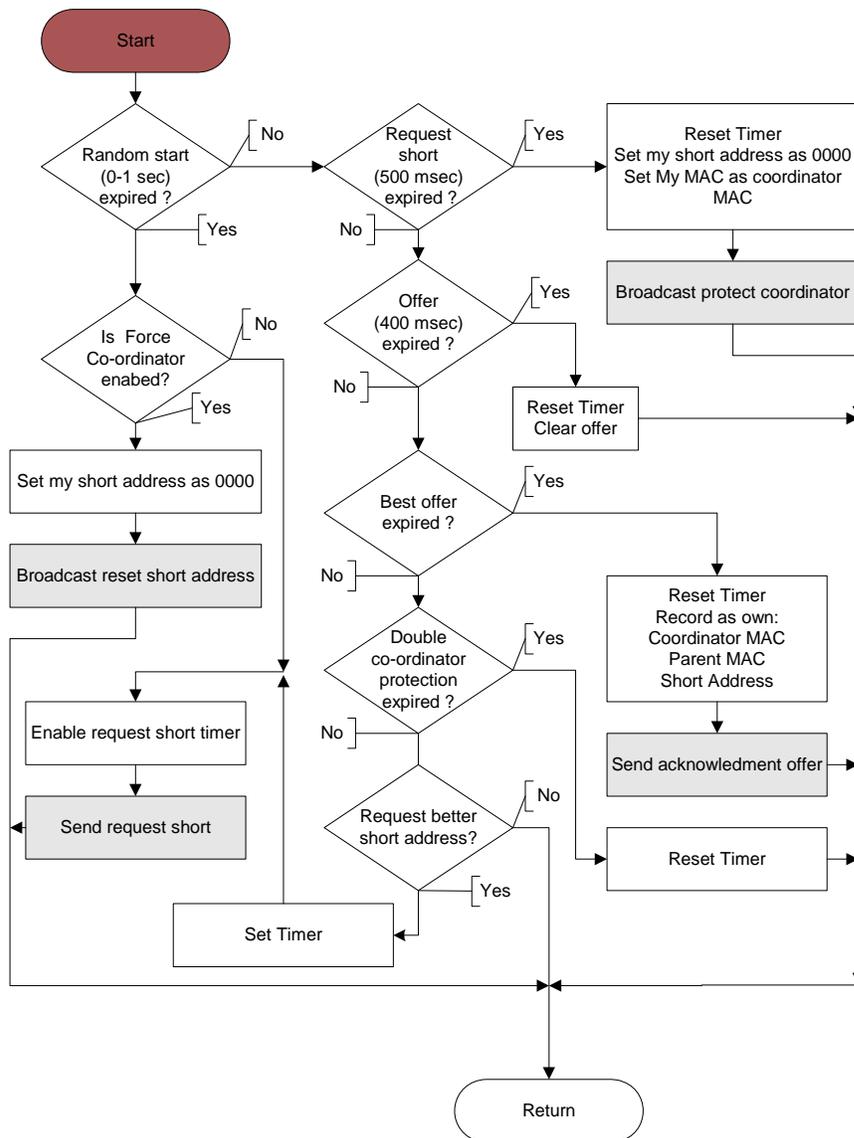


Figure 4.18 Flow diagram for short address timers

This technique protects the wireless network when devices need to request new short addresses simultaneously. This will occur when non-battery powered devices experience a power failure or when a Reset coordinator command was sent after a second coordinator was discovered in the WPAN.

The Force coordinator functionality enables a device to always become the coordinator. This functionality can be used when the device is used for example to monitor all alarms from a nerve centre or when a device is equipped with the most powerful processor.

If the Force coordinator functionality is enabled, the device will set its short address as 0x0000. It will either broadcast a Reset short address command to all devices or it will enable the Request short address timer and send a Request short address.

- *Request short address expires*

When a device requests a short address as described above, the Request short address timer is enabled. The timer is disabled when the device receives an offer back as described in section 4.5.2.4. If no offer is received in the pre-defined period the timer will expire and the device will then elect itself as the coordinator and disable the timer. A Protect coordinator will be broadcasted after a normal election.

- *Offer expires*

When a device receives a request for a short address it will reply with an offer and enable the Offer timer as described in section 4.5.2.3. If the offer is not accepted in a pre-defined time the timer will expire and the device will clear the offer.

- *Best offer expires*

After a device had requested a short address and it receives an offer, it will enable the Best offer expire timer or reset the timer if a better offer is received as discussed in section 4.5.2.3. When the timer expires, the device will use the short address that was previously recorded as the best short address offer. It will then send an acknowledgment to the device that sent the Best offer.

- *Double coordinator protection*

When a device becomes a new coordinator, it will send a Protect coordinator command to inform other devices that a new coordinator was elected. This will disable the Reset short address command for a pre-defined time. In this period all devices will ignore the inspection for double coordinators. This was discussed in section 4.5.2.3.

The old coordinator will disappear in the device's parent-child relationship tables after the Keepalive timers had expired. When the timer expires all devices will protect any changes in the coordinator field.

- *Request better short address*

When a device receives a short address, it must periodically request a better short address. In the simulation it is set to two minutes. This mechanism will ensure that the Tree structure is always updated. Device 0x000 will never request a better address; firstly, because it is the coordinator and secondly it already has the best address in the PAN.

An example is that a device is in range of Device 0x0100 and 0x0200. Device 0x0100 already has the maximum amount of children. Our device short address will be 0x0210. Every two minutes the device will request a better short address, but Device 0x0200 will keep offering 0x0210. Device 0x0140 is now removed. When Device 0x0210 requests a better short address, Device 0x0100 will offer 0x0140 and Device 0x0200 will offer 0x0210. Device 0x0210 will then change its short address to 0x0140.

Another example is when three devices are all children of Device 0x0000. Their short addresses are 0x0100, 0x0200 and 0x0300 respectively. Device 0x0200 is removed. When Device 0x0300 now requests a better short address, it is offered 0x0200.

## 4.6 Keepalive function development

The Keepalive function is used to maintain the topology. It must ensure that all parent-child tables are updated.

### 4.6.1 Frame format

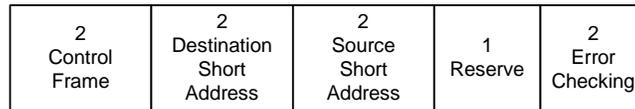


Figure 4.19 Keepalive frame

Figure 4.19 illustrates the Keepalive frame. The number above each block indicates the length in bytes for each field. Bit 1 of the first control field identifies the Keepalive frame. The remote devices short address can be calculated from the position entry in the parent-child table as illustrated in Table 4.2. Entry 0x0 is reserved for the parent and 0x1 to 0xE for the device's children.

Table 4.2 Parent-child relationship table

Entry	MAC	Keepalive timer
0x0	0x0000000000000000	0
:	:	:
0xE	0x0000000000000000	0

If the device's short address is 0x0000, entry 0x0's MAC address will always be 0x0000000000000000. The short addresses of the device's children will be 0x0100 for entry 0x1 and 0x0E00 for 0xE.

If the device's short address is 0x0300, entry 0x0's MAC address will be from device 0x0000. The MAC address for entry 0x1 will be for device 0x0310 and 0xE for device 0x0E00.

If the device short address is 0x0420, entry 0x0's MAC address will be from device 0x0420's parent, 0x0400. The MAC address for entry 0x1 will be for device 0x0421 and 0xE for device 0x0E00.

If the devices short address is 0x0731, only entry 0x0 will have a valid MAC address. Entry 0x0 to 0xE will always be 0x0000000000000000.

The Keepalive suite consists of two commands. The functionality of the commands was discussed in section 3.2.2.2.

- Request Keepalive



Figure 4.20 Request Keepalive frame sample

- Acknowledge Keepalive

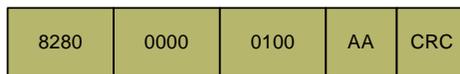


Figure 4.21 Acknowledge Keepalive frame sample

The two commands always function together. In a parent-child relationship both devices will have a Keepalive timer for that relationship. Figure 4.22 illustrates the simulator's parent-child relationship register for devices 0x0000 and 0x0100. Register 0x0 has been called parent and only nine children are allowed.

Register	Value	Time
Device	0000001641A75694	6
Coordinator	0000001641A75694	
Parent	0000000000000000	0
1	000000065BC31340	0
2	0000000000000000	0
3	0000000000000000	0
4	0000000000000000	0
5	0000000000000000	0
6	0000000000000000	0
7	0000000000000000	0
8	0000000000000000	0
9	0000000000000000	0
Short	0000	
Short Offer	FFFF	
MAC1		0
MAC2		0

(a)

Register	Value	Time
Device	000000065BC31340	0
Coordinator	0000001641A75694	
Parent	0000001641A75694	5
1	0000000000000000	0
2	0000000000000000	0
3	0000000000000000	0
4	0000000000000000	0
5	0000000000000000	0
6	0000000000000000	0
7	0000000000000000	0
8	0000000000000000	0
9	0000000000000000	0
Short	0100	
Short Offer	FFFF	
MAC1		0
MAC2		0

(b)

Figure 4.22 Parent-child relationship table view on simulator

The simulator also shows the device and its coordinator's MAC address, its short address and if there is any outstanding short address offers. One device's timer will always be lower than the other device; when the child accepts the address, it will start its timer for that relationship. The parent will only start its timer after the acknowledgement was received. For this reason one device will always send the Request Keepalive before the other device. The other device will reset its counter and send an Acknowledge Keepalive. The original sender will reset its counter when the acknowledgement is received. A Request Keepalive will be sent every ten seconds and if the counter was not cleared in fifteen seconds the relationship is cleared.

## 4.6.2 Algorithms

### 4.6.2.1 Keepalive received

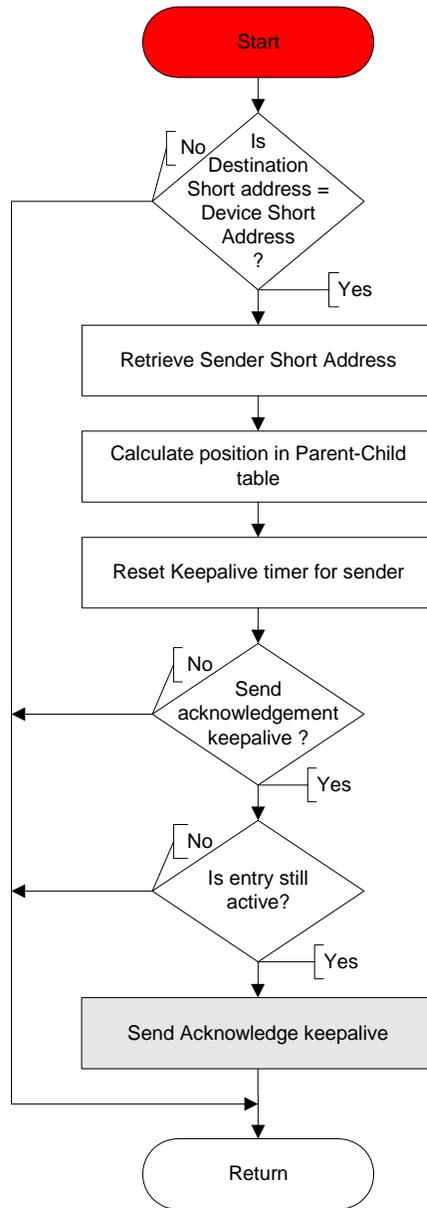


Figure 4.23 Flow diagram for Keepalive received

The device only accepts a Keepalive frame that has the device's short address in the Destination short address field. The sender's short address is recorded and used to calculate the position in the parent-child relationship table. That specific counter is then reset to zero. If the frame received was an Acknowledge Keepalive frame the procedure can terminate, otherwise an Acknowledge Keepalive reply must be sent. Before an Acknowledge Keepalive frame is sent the device will first verify that the entry in the parent-child relationship table is still valid.

#### **4.6.2.2 Keepalive timer**

The Keepalive timer is part of the global timer process and must be seen as part of the other timer procedures. The timers are used to send periodic Keepalive frames and the timer consist of two parts:

- Maintain the counters

Every second all entries where the MAC address is not 0x000000000000, is incremented by one.

- Sending Keepalives

Every 500 milliseconds the device will investigate the counters to determine if any Request Keepalive frames must be sent. The device will only send one Keepalive at a time. This technique is used to protect the wireless spectrum against congestion. The simulator was developed to only have one parent and nine children, even if the standard allows for fifteen children.

Table 4.3 *Keepalive time schedule*

Keepalive	Schedule							
	0 ms	:	500 ms	600 ms	700 ms	800 ms	900 ms	1000 ms
0 - 9 sec								
10 sec				1	2	3	4	5
11 sec				6	7	8	9	0
12 sec				1	2	3	4	5
13 sec				6	7	8	9	0
14 sec				1	2	3	4	5
15 sec				6	7	8	9	0
16 sec					Clear relationship			

Table 4.3 illustrates the time intervals that devices will send Keepalive frames to its parent or children. Zero to nine represents the entry position in the parent-child relationship table. It was decided to only send Keepalive frames in the second half of a second. This allows the processor to provide other services in the first half of a second. Only one Request Keepalive is sent out in a 100 ms interval.

Figure 4.24 illustrates the procedure for the device to send three Request Keepalives attempts before it will remove the relationship between the two devices. The position in the parent-child relationship table will increment every 100 ms, if the counter is 500 ms or higher. If the Keepalive counter is higher than fifteen the entry will be cleared, but if the counter is higher than ten the device will send out a Request Keepalive every two seconds. After three attempts the counter will be sixteen and the entry will be cleared.

If the parent relation is cleared all relations to all children are also cleared and the device will clear its short address and start the process to request a new short address. All children now need to request new short addresses.

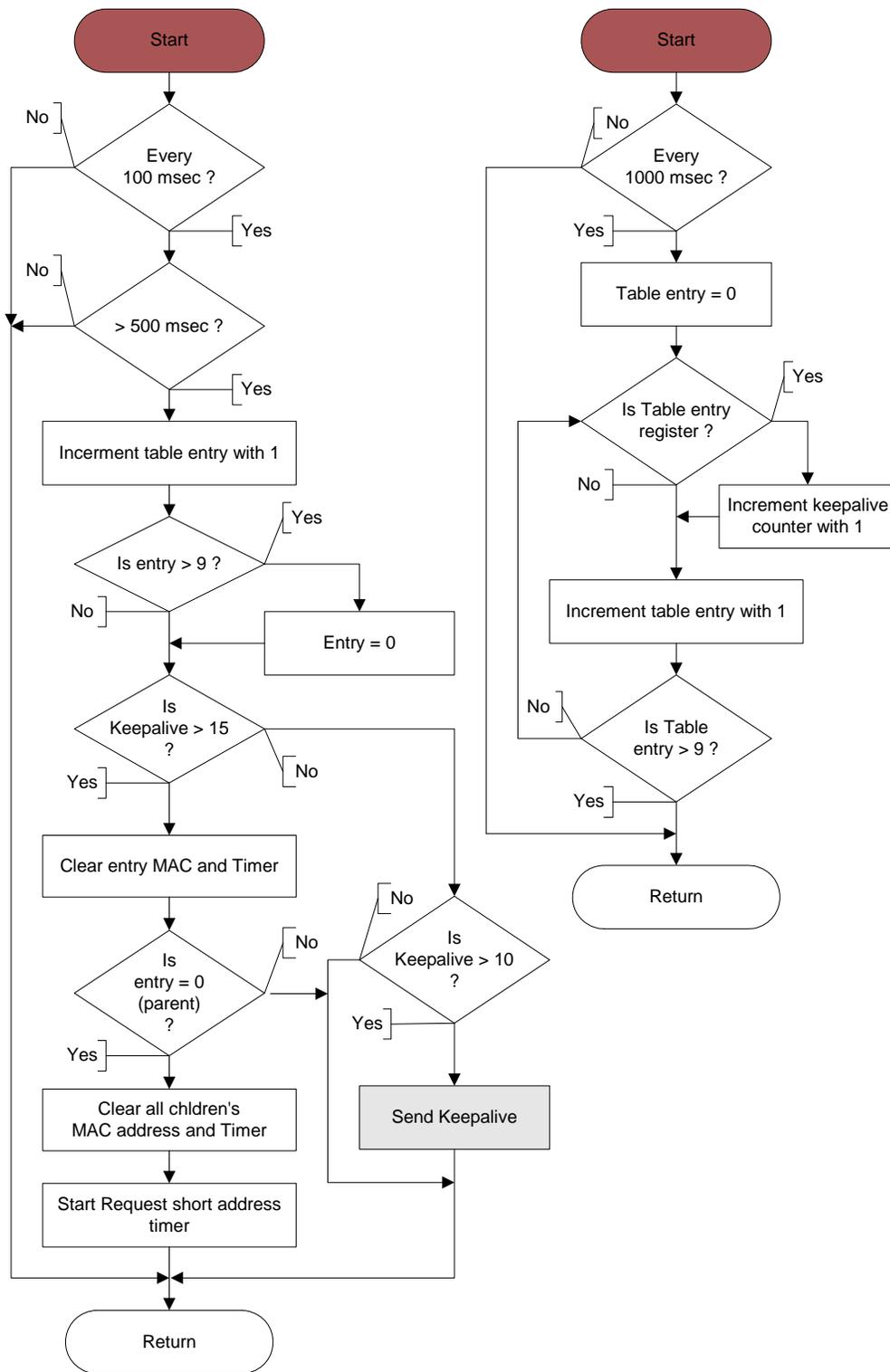


Figure 4.24 Flow diagram for Keepalive timer

## 4.7 Simulation and results

The trace tab that was discussed in section 4.2.3 was used to capture all data. A range of simulations was done to prove the functionality of the algorithms decrypted in this chapter.

### 4.7.1 Three devices request short addresses in same broadcast domain

In the first simulation three devices are setup to be in the same broadcast domain. Devices are powered on in sequence from Device 1 to Device 3. Device 1 to Device 3's packet captures is shown from Table 4.4 to Table 4.6. All transmitted packets are indicated in blue and received in red.

Table 4.4 Request short address. Device 1: MAC 0000001641A75694

Step	Device 1 : MAC 0000001641A75694
1	8800FFFFFFFFFFFFFFFF0000001641A75694FFFFFFFFFFFFFFFFFAFFFF5A07
2	8820FFFFFFFFFFFFFFFF0000001641A75694FFFFFFFFFFFFFFFFFAFFFF8803
3	8800FFFFFFFFFFFFFFFF000000065BC31340FFFFFFFFFFFFFFFFFAFFFF41B5
4	8840000000065BC31340000001641A756940000001641A75694AA0100B64E
5	88800000001641A75694000000065BC31340000001641A75694AA0100182A
6	8800FFFFFFFFFFFFFFFF0000000874B38C2EFFFFFFFFFFFFFFFFFAFFFF63FA
7	
8	88400000000874B38C2E0000001641A756940000001641A75694AA0200BD57
9	88800000001641A756940000000874B38C2E0000001641A75694AA0200EF36
10	820001000000AA8425
11	828000000100AACD64
12	820002000000AA6AF7
13	828000000200AA9434

Table 4.5 Request short address. Device 2: MAC 000000065BC31340

Step	Device 2: MAC 000000065BC31340
1-2	Powered Down
3	8800FFFFFFFFFFFFFFFF000000065BC31340FFFFFFFFFFFFFFFFFAFFFF41B5
4	8840000000065BC31340000001641A756940000001641A75694AA0100B64E
5	88800000001641A75694000000065BC31340000001641A75694AA0100182A
6	8800FFFFFFFFFFFFFFFF0000000874B38C2EFFFFFFFFFFFFFFFFFAFFFF63FA
7	88400000000874B38C2E000000065BC31340000001641A75694AA0110E187
8	
9	Offer expired
10	820001000000AA8425
11	828000000100AACD64
12-13	

Table 4.6 Request short address. Device 3: MAC 000000874B38C2E

Step	Device 3: MAC 000000874B38C2E
1 - 5	Powered Down
6	8800FFFFFFFFFFFFFFFF000000874B38C2EFFFFFFFFFFFFFFFFAFFFFFF63FA
7	8840000000874B38C2E00000065BC31340000001641A75694AA0110E187
8	8840000000874B38C2E0000001641A756940000001641A75694AA0200BD57
9	88800000001641A75694000000874B38C2E0000001641A75694AA0200EF36
10 -11	
12	820002000000AA6AF7
13	828000000200AA9434

The following happened at each step in Table 4.4 to Table 4.6:

1. Device 1 transmits a Request short address packet.
2. Device 1 does not receive any offer and becomes the coordinator. It sends a protect coordinator packet.
3. Device 2 transmits a Request short address. Only Device 1 receives the request.
4. Device 1 sends an offer 0x0100 back to Device 2.
5. Device 2 did not receive a better offer and accepts the short address.
6. Device 3 is powered on and requests a short address.
7. Device 2's offer was received first.
8. Device 1's offer follows.
9. Device 3 accepts the best offer from Device 1 and the offer from Device 2 times out.
10. The Keepalive timer for Device 2 to its parent expires and a Request Keepalive from short address 0x0100 is sent to 0x0000. Device 1 resets its Keepalive timer for child 0x0100.
11. Device 1 returns an Acknowledge Keepalive to Device 2 with short address 0x0100. Device 2 resets its Keepalive timer for its parent.
12. The Keepalive timer for Device 3 to its parent expires and a Request Keepalive from short address 0x0200 is sent to 0x0000. Device 1 resets its Keepalive timer for child 0x0200.
13. Device 1 returns an Acknowledge Keepalive to Device 3 with short address 0x0200. Device 2 resets its Keepalive timer for its parent.

The Keepalive sequence from step 9 to step 12 will repeat itself until a neighbour ship is lost or when a new device joins the PAN.

#### 4.7.2 Cleanup parent-child table

In this simulation, Device 1 of the previous simulation is powered down and its short address 0x0100 is reoffered to Device 2. The devices packet captures are shown in Table 4.7 and Table 4.8.

Table 4.7 Cleanup parent-child table. Device 1: MAC 0000001641A75694

Step	Device 1 : MAC 0000001641A75694
1	820001000000AA8425
2	820001000000AA8425
3	820001000000AA8425
4	820002000000AA6AF7
5	828000000200AA9434
6	Steps 5 to 6 repeats for 2 minutes
7	8800FFFFFFFFFFFFFFFF000000874B38C2E0000001641A75694AAFFFFEA07
8	8840000000874B38C2E0000001641A756940000001641A75694AA0100E804
9	88800000001641A75694000000874B38C2E0000001641A75694AA0100BA75

Table 4.8 Cleanup parent-child table. Device 3: MAC 0000000874B38C2E

Step	Device 3: MAC 0000000874B38C2E
1 - 3	
4	820002000000AA6AF7
5	828000000200AA9434
6	Steps 5 to 6 repeats for 2 minutes
7	8800FFFFFFFFFFFFFFFF000000874B38C2E0000001641A75694AAFFFFEA07
8	8840000000874B38C2E0000001641A756940000001641A75694AA0100E804
9	88800000001641A75694000000874B38C2E0000001641A75694AA0100BA75

The following happens at each step in Table 4.7 and Table 4.8:

1. Device 2 was powered down and its parent sends a Request Keepalive after the Keepalive timer reaches 10 seconds.
2. No reply after 2 seconds from Device 2 and another Request Keepalive is sent.

3. No reply after another 2 seconds from Device 2 and another Request Keepalive is sent. When the timer counter reaches 15 seconds the relationship between Device 1 and Device 2 is broken. The entry for 0x0100 at device 0x0000 is now spare.
4. Device 1 sends a Request Keepalive to Device 3 with short address 0x0200.
5. Device 3 acknowledges the Request Keepalive.
6. Steps 4 to 5 are repeated for 2 minutes until the Discover better short address timer expires.
7. Device 3 with short address 0x0200, requests a better short address. Device 1 clears any parent-child entry for MAC 0x0000000874B38C2E.
8. Device 1 now offers 0x0100 to Device 2.
9. Device 2 does not receive any better offers and accepts offer 0x0100.

### 4.7.3 New coordinator selected

This simulation is the same as in 4.7.2, but now Device 1, the coordinator, is powered down. The devices packet captures are shown in Table 4.9 and Table 4.10.

Table 4.9 New coordinator selected. Device 2: MAC 000000065BC31340

Step	Device 2: MAC 000000065BC31340
1-2	
3	820000000100AA1944
4	
5	820000000100AA1944
6	8800FFFFFFFFFFFFFFFF0000000874B38C2EFFFFFFFFFFFFFFFFFAAFFFE3FA
7	88400000000874B38C2E000000065BC31340000001641A75694AA0110E187
8	8880000000065BC31340000000874B38C2E0000001641A75694AA0110D1D8
9	820000000100AA1944
10	8800FFFFFFFFFFFFFFFF000000065BC31340FFFFFFFFFFFFFFFFFAAFF41B8
11	8820FFFFFFFFFFFFFFFF000000065BC31340FFFFFFFFFFFFFFFFFAA0008EBE
12-14	
15	8800FFFFFFFFFFFFFFFF0000000874B38C2EFFFFFFFFFFFFFFFFFAAFFFE3FA
16	88400000000874B38C2E000000065BC3134000000065BC31340AA01000343
17	8880000000065BC31340000000874B38C2E000000065BC31340AA0100331C

Table 4.10 New coordinator selected. Device 3: MAC 0000000874B38C2E

Step	Device 3: MAC 0000000874B38C2E
1	82000000200AA4014
2	82000000200AA4014
3	
4	82000000200AA4014
5	
6	8800FFFFFFFFFFFFFFFF000000874B38C2EFFFFFFFFFFFFFFFFFAAFFFE3FA
7	8840000000874B38C2E00000065BC31340000001641A75694AA0110E187
8	888000000065BC3134000000874B38C2E0000001641A75694AA0110D1D8
9	
10	8800FFFFFFFFFFFFFFFF00000065BC31340FFFFFFFFFFFFFFFFFAAFF41B8
11	8820FFFFFFFFFFFFFFFF00000065BC31340FFFFFFFFFFFFFFFFFAA00008EBE
12	82001000110AAB066
13	82001000110AAB066
14	82001000110AAB066
15	8800FFFFFFFFFFFFFFFF000000874B38C2EFFFFFFFFFFFFFFFFFAAFFFE3FA
16	8840000000874B38C2E00000065BC313400000065BC31340AA01000343
17	888000000065BC3134000000874B38C2E00000065BC31340AA0100331C

The following happens at each step in Table 4.9 and Table 4.10:

1. Device 3's, with short address 0x0200, Keepalive timer for its parent expires and it sends a Request Keepalive.
2. Device 3 did not receive an Acknowledge Keepalive back after two seconds and sends a second Request Keepalive.
3. Device 2's, with short address 0x0100, Keepalive timer for its parent expires and it sends a Request Keepalive.
4. Device 3 did not receive an Acknowledge Keepalive back after two seconds and sends a last Request Keepalive.
5. Device 2 did not receive an Acknowledge Keepalive back after two seconds and sends a second Request Keepalive.
6. Device 3 did not receive an Acknowledge Keepalive back and clears its short address. After a random time it requests a new short address.
7. Device 2 still has short address 0x0100 and offers short address 0x0110.
8. Device 3 accepts the only offer 0x0110.

9. Device 2 did not receive an Acknowledge Keepalive back after two seconds and sends a last Request Keepalive.
10. Device 2 did not receive an Acknowledge Keepalive back and clears its short address. After a random time it requests a new short address. Device 3 will not offer a short address to Device 2 because its MAC address is still recorded as its Device 3's parent.
11. Device 2 does not receive any short address offers and it elects itself as the coordinator and sends a Protect coordinator to all children.
12. Device 3's, with short address 0x0110, Keepalive timer for its parent expires and it sends a Request Keepalive.
13. Device 3 did not receive an Acknowledge Keepalive back after two seconds and sends a second Request Keepalive.
14. Device 3 did not receive an Acknowledge Keepalive back after two seconds and sends a last Request Keepalive.
15. Device 3 did not receive an Acknowledge Keepalive back and clear its short address. After a random time it requests a new short address.
16. Device 2 offers short address 0x0100.
17. Device 3 accepts short address 0x0100.

#### **4.7.4 Two coordinators detected**

In this simulation three devices were setup. The first two devices, Device 2 and 3, start up normally.

To simulate a second coordinator the third simulator is setup as follows:

- select force coordinator, and
- deselect new coordinator protection.

These tabs are on the global page shown in Figure 4.2. This is only to simulate a second coordinator and by default the protocol will not allow the above. After Device 1 elects itself as the coordinator,

the New coordinator protection tab must be selected before the child requests a better short address and discovers two coordinators.

Table 4.11 Two coordinators detected. Device 1: MAC 0000001641A75694

Step	Device 1 : MAC 0000001641A75694
1	Coordinator
2	8800FFFFFFFFFFFFFFFF000000065BC31340000000874B38C2EAAFFFF3FD4
3	
4	8840000000065BC31340000001641A75694000001641A75694AA0100B64E
5	88200000001641A7569400000065BC31340FFFFFFFFFFFFFFFFFAAFFFF959E
6	8800FFFFFFFFFFFFFFFF0000001641A75694FFFFFFFFFFFFFFFFFAAFFFF5A07
7	Becomes 0x0200

Table 4.12 Two coordinators detected. Device 2: MAC 000000065BC31340

Step	Device 2: MAC 000000065BC31340
1	Child of Device 3
2	8800FFFFFFFFFFFFFFFF000000065BC31340000000874B38C2EAAFFFF3FD4
3	8840000000065BC31340000000874B38C2E0000000874B38C2EAA0100782F
4	8840000000065BC313400000001641A75694000001641A75694AA0100B64E
5	88200000001641A7569400000065BC31340FFFFFFFFFFFFFFFFFAAFFFF959E
6	8800FFFFFFFFFFFFFFFF0000001641A75694FFFFFFFFFFFFFFFFFAAFFFF5A07
7	Offer 0x0110

Table 4.13 Two coordinators detected. Device 3: MAC 0000000874B38C2E

Step	Device 3: MAC 0000000874B38C2E
1	Coordinator and parent of Device 2
2	8800FFFFFFFFFFFFFFFF000000065BC31340000000874B38C2EAAFFFF3FD4
3	8840000000065BC31340000000874B38C2E0000000874B38C2EAA0100782F
4-5	
6	8800FFFFFFFFFFFFFFFF0000001641A75694FFFFFFFFFFFFFFFFFAAFFFF5A07
7	Offer 0x0200

The following happens at each step in Table 4.11 to Table 4.13:

1. Device 3 is the parent and coordinator for Device 2. Device 1 is incorrectly also a coordinator and there are no children devices of it.
2. Device 2's Discover better short address timer expired and it sends a Request short address.
3. Device 3's offer is received first and Device 2 waits for better offers.
4. Device 1's offer is received. Device 2 notices that the second offer coordinator MAC address is different then the previous best offer MAC address.

5. Device 2 sends a Double coordinator command to the last offer. In this simulation it was to the device that has the incorrect address, but this is just because it was the last offer.
6. Device 1 clears its short address and requests a new short address.
7. The same process now occurs that was discussed in 4.7.1 from step 6. It receives a 0x0110 and 0x0200 offer and accepts 0x0200.

#### 4.7.5 New coordinator forced

In this simulation Device 2 and 3 starts up normally. Afterwards Device 1 is powered up and is configured to always be the coordinator. The Force coordinator is selected as shown in Figure 4.2 for Device 1.

Table 4.14 Force coordinators. Device 1: MAC 0000001641A75694

Step	Device 1 : MAC 0000001641A75694
1	
2	8820FFFFFFFFFFFFFFFFF0000001641A75694FFFFFFFFFFFFFFFFFAAFF8803
3	8800FFFFFFFFFFFFFFFFF000000065BC31340FFFFFFFFFFFFFFFFFAAFF41B5
4	884000000065BC31340000001641A756940000001641A75694AA0100B64E
5	88800000001641A75694000000065BC31340000001641A75694AA0100182A
6	8800FFFFFFFFFFFFFFFFF0000000874B38C2EFFFFFFFFFFFFFFFFFAAFFE3FA
7	
8	8840000000874B38C2E0000001641A756940000001641A75694AA0200BD57
9	88800000001641A756940000000874B38C2E0000001641A75694AA0200EF36

Table 4.15 Force coordinators. Device 2: MAC 000000065BC31340

Step	Device 2: MAC 000000065BC31340
1	Child of Device 3
2	8820FFFFFFFFFFFFFFFFF0000001641A75694FFFFFFFFFFFFFFFFFAAFF8803
3	8800FFFFFFFFFFFFFFFFF000000065BC31340FFFFFFFFFFFFFFFFFAAFF41B5
4	884000000065BC31340000001641A756940000001641A75694AA0100B64E
5	88800000001641A75694000000065BC31340000001641A75694AA0100182A
6	8800FFFFFFFFFFFFFFFFF0000000874B38C2EFFFFFFFFFFFFFFFFFAAFFE3FA
7	8840000000874B38C2E000000065BC31340000001641A75694AA0110E187
8-9	

Table 4.16 Force coordinators. Device 3: MAC 000000874B38C2E

Step	Device 3: MAC 000000874B38C2E
1	Coordinator and parent of Device 2
2	8820FFFFFFFFFFFFFFFF0000001641A75694FFFFFFFFFFFFFFFFFAAFFFF8803
3-5	
6	8800FFFFFFFFFFFFFFFF000000874B38C2EFFFFFFFFFFFFFFFFFAAFFFFE3FA
7	8840000000874B38C2E00000065BC31340000001641A75694AA0110E187
8	8840000000874B38C2E0000001641A756940000001641A75694AA0200BD57
9	88800000001641A75694000000874B38C2E0000001641A75694AA0200EF36

The following happens at each step in Table 4.14 to Table 4.16:

1. Device 3 is the coordinator and parent of Device 2.
2. Device 1 starts up, elects itself as the coordinator and broadcasts to all devices to clear their short addresses and request, in random times, new short addresses.
3. Device 2 first requests a short address. Device 3 will not accept any packets because it does not have a short address.
4. Device 1 offers 0x0100.
5. Device 2 accepts offer 0x0100.
6. Device 3's random start timer expires last and it requests a new short address.
7. Device 3 receives Device 2's offer 0x0110 first.
8. Device 1's offer 0x0200 is received second, but still before the Discover better offer timer expires.
9. Device 3 accepts offer 0x0200.

## 4.8 Conclusion

The only responsibility of the short address and Keepalive commands is to maintain the physical layout of the devices. The address structure is hierarchical. Any device can be the coordinator, parent or child. Devices can function as a parent to other devices and be a child of another device. The coordinator can also function as a parent. The only setup needed is, if required, to configure one device to always be the coordinator.

# Chapter 5

## 5 Maintaining connections between control points

This chapter will discuss the process of how a connection is established and maintained between device's endpoints. This suite will function above the Discovery suite and is not responsible for maintaining any short addresses.

### 5.1 Connection frames

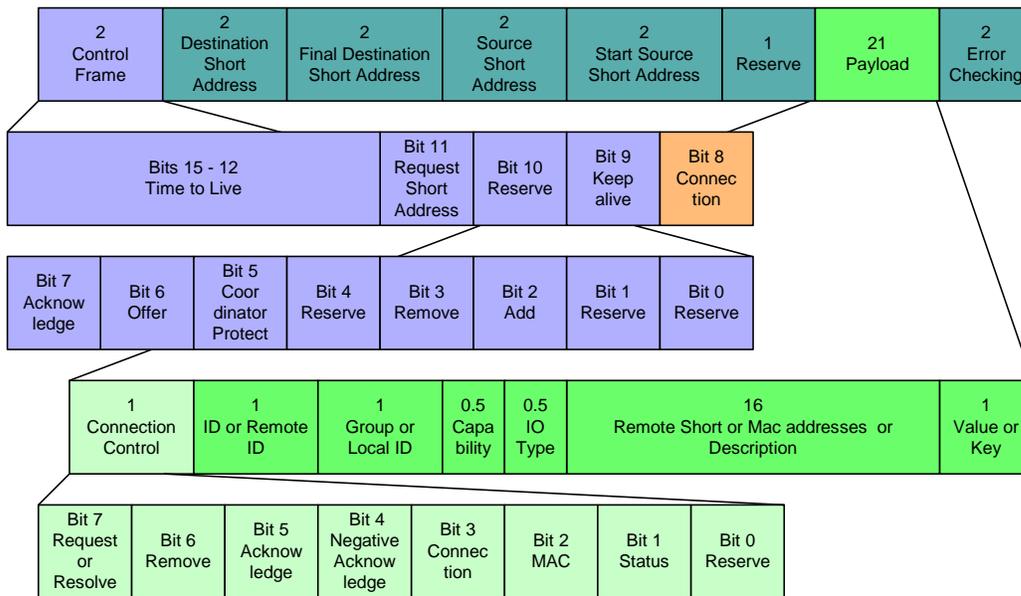


Figure 5.1 Connection frame format

Figure 5.1 illustrates the Connection frame. The number above each field indicates the byte length of the field, one byte equals to eight bits. A detailed view of both control frames is also included.

A connection frame consists of two parts:

- *Routing*

The first field, the control frame, is the same as in the Discovery suite and was discussed in Chapter 4. Bit 0 of the Control frame field indicate that this frame is a Connection frame and its value is always 0x8100 for a Connection frame. A Connection frame is routable between multiple devices.

The devices were designed to function without routing tables. Section 3.2.3.1 discussed the functionality of the four address fields, but in brief the device records its short address as the source address and the remote device, hosting the remote endpoint, as the destination address. The forward source and destination fields are changed at each device. Each device, including the source device, will record itself as the forward source address and calculate the forward destination address. Routing can be done via addresses, because the topology is a hierarchical structure. The only forward decision a device needs to calculate is if the packet must be forwarded to a child or to its parent.

The error-checking field uses the same mechanism as the Discovery suite, namely CRC-16-CCITT and it was discussed in section 2.3.

- *Connection*

The connection portion has its control field, namely the Connection control field. Each field inside the Connection control frame will be explained with each command in the following sections. The following functions of the six fields will change according to the Control connection field value.

When an endpoint needs to setup a connection to another endpoint the following occurs:

1. An endpoint sends a request connection to the coordinator. It specifies its ID number, group, capabilities, IO type and key.
2. The same occurs at the remote endpoint. When two requests have the same key, the coordinator will compare their groups, capabilities and IO types.
3. If the requests were unsuccessful a Negative acknowledge connection will be sent from the coordinator to both hosting devices and remove the connection request from its table.
4. If the two requests are compatible with each other, the coordinator will send the information of the endpoint to the hosting device via an Acknowledge connection command and remove the connection request from its table.
5. Before the coordinator sends an acknowledgement or unacknowledgement, a Resolve MAC will be sent to both devices to confirm that both devices short addresses have not changed.
6. If the device does not get an Acknowledge MAC back from the device, the coordinator will broadcast a Request MAC. Point 1 to 6 has been previously illustrated in Figure 3.19.
7. Both devices will now update each other with their current status.
8. The device will send a Status command to the Remote device when the status of an endpoint changes. The status will also be updated periodically as previously illustrated in Figure 3.21.
9. When an endpoint connection is terminated, the hosting device will send a Remove connection to the remote device after the short address was confirmed.

## **5.2 Forward function development**

The forward function is part of all the Connection commands. Before any Connection command is executed, the receiving device will investigate the destination short address to determine if the frame is at its destination or if it must be forwarded.

## 5.2.1 Algorithms

Figure 5.2 illustrates the algorithm for the forward function. The device first investigates the forward destination short address when a Connection frame is received. The device will ignore the frame if it is not a broadcast or the short address of the device.

The Connection command will be executed, if the destination short address of the frame is the same as the short address of the device. This is discussed in section 5.3 to 5.5.

The device needs to forward the frame if the destination short address is not the short address of the device. The frames TTL is decreased at each hop and the frame are dropped when the value becomes zero. To limit the Keepalive process to request better short addresses, the Request better short address is delayed with five seconds if the timer expires in less than five seconds. This will prevent the device requesting a better short address while it is busy with a transaction. The next hop is calculated and if the device short address is 0x0100 and the Destination short address is 0x0122 then the device will forward the frame to device 0x0120. The next hop calculation was previously explained in section 3.2.3.1.

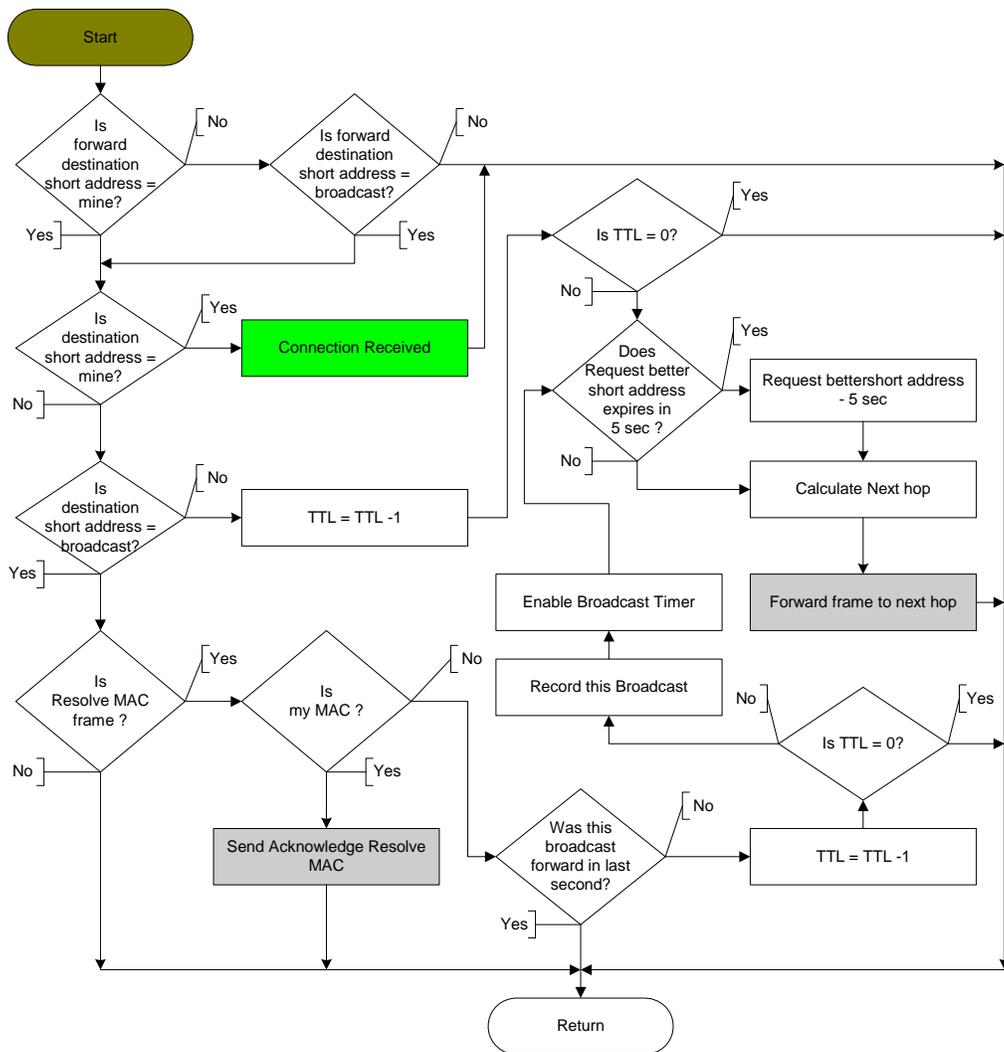


Figure 5.2 Flow diagram for Connection forward

If the destination short address is a Resolve MAC Broadcast and the MAC address belongs to the device an Acknowledge Resolve MAC frame is returned to the original requesting device otherwise this broadcast needs to be forwarded to all neighbouring devices.

As explained in section 3.2.3.3 under Figure 3.23 a mechanism must prevent a broadcast storm. The solution is that the device records the broadcast frame before it forwards the broadcast and will not forward the same frame in the next second. The TTL value of the frame and the Request better short address timer are checked before the broadcast frame is sent.

### **5.3 Resolve Medium Access Control address function development**

When a connection is setup between two endpoints the MAC address of the remote hosting device is recorded. The Connection command only communicates to other devices with their short addresses. If the short address of a device changes, all its remote endpoints need to discover the new short address. The Resolve Medium Access Control address suite is used to resolve a MAC address to a short address. It consists of a Resolve MAC and an Acknowledge MAC address command. When a connection is established, the coordinator also uses the Resolve MAC command to confirm the short addresses before it hands over the connection to the hosting devices. This will be discussed later in this chapter.

The Resolve MAC address command has two variations. The first is where the command is sent to a known destination also known as a unicast transmission [15] and the second where the command is broadcasted to all devices. When one device must confirm the short address of another device, it will transmit a unicast Resolve MAC address to the device. If the requesting device does not receive an Acknowledge MAC address reply in a pre-defined period, a broadcast Resolve MAC address is transmitted. If an Acknowledge MAC address is still not received in the pre-defined period, the requesting device will enable another timer to allow it to repeat the above process until an acknowledgement is received or until the connection is terminated.

If a device receives a Resolve MAC address request and the specified MAC address does not belong to the specified short address, a Negative Acknowledge MAC address command is returned.

### 5.3.1 Frame format

Only the payload of the frame in Figure 5.1 changes and this is illustrated in Figure 5.3. In the MAC suite frame, the second to fourth and last bytes, in the payload, are reserved. Bit 2 of the Connection control field indicates any MAC suite command.

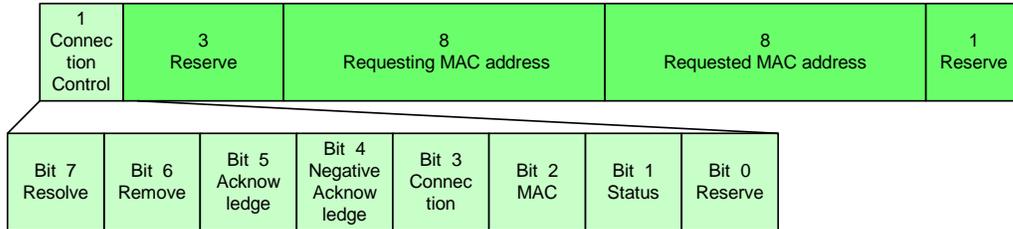


Figure 5.3 Resolve Medium Access Control address frame

- Resolve Medium Access control address

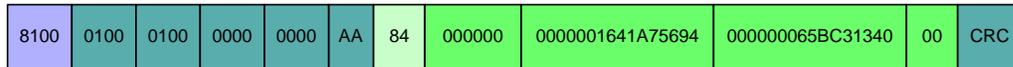


Figure 5.4 Unicast Resolve Medium Access Control frame sample

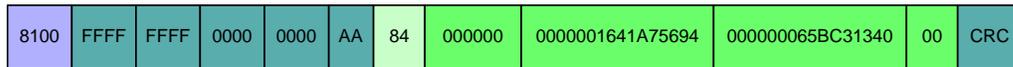


Figure 5.5 Broadcast Resolve Medium Access Control frame sample

- Acknowledge Medium Access Control address

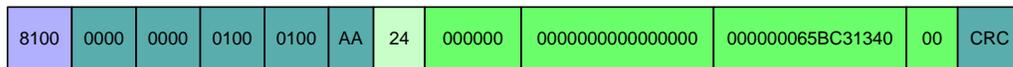


Figure 5.6 Acknowledge Medium Access Control address frame sample

- Negative Acknowledge Medium Access Control address

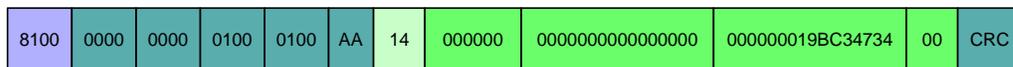


Figure 5.7 Negative Acknowledge Medium Access Control address frame sample

Figure 5.4 and Figure 5.5 illustrates when the coordinator, with MAC address 0x0000001641A75694, requests Device B, with short address 0x0100 and MAC address 0x000000065BC31340, to acknowledge its short address. The Resolve MAC address frame specifies the local and remote MAC. The Acknowledge MAC address only specifies the MAC address of the local device, as illustrated in Figure 5.6. This will happen when a device receives a Resolve MAC address command, it will update all the short addresses of established connections associated with the specific MAC address, before an Acknowledge MAC address command is send. The requesting device will update the short addresses of its connections with the short addresses of the source that has been specified in the Acknowledge MAC address replies.

Figure 5.7 illustrates the Negative acknowledge MAC address command that is sent if Device A, with short address 0x0100, did receive a unicast Resolve MAC request command and its MAC address is not 0x000000065BC31340.

### **5.3.2 Algorithms**

Two techniques must be explained in order to understand the MAC address resolve algorithms.

Firstly, the device must be able to resolve two MAC addresses simultaneously. This is necessary when the coordinator found a partner for a pending request connection. (This will be discussed later in the section about manage connections between endpoints in section 5.4.) This is also necessary when the IO of an endpoint is an “OR” or “AND” and the status of the local endpoint needs to be updated on two divert devices. (This will be discussed in the section about manage endpoint statuses in section 5.5.)

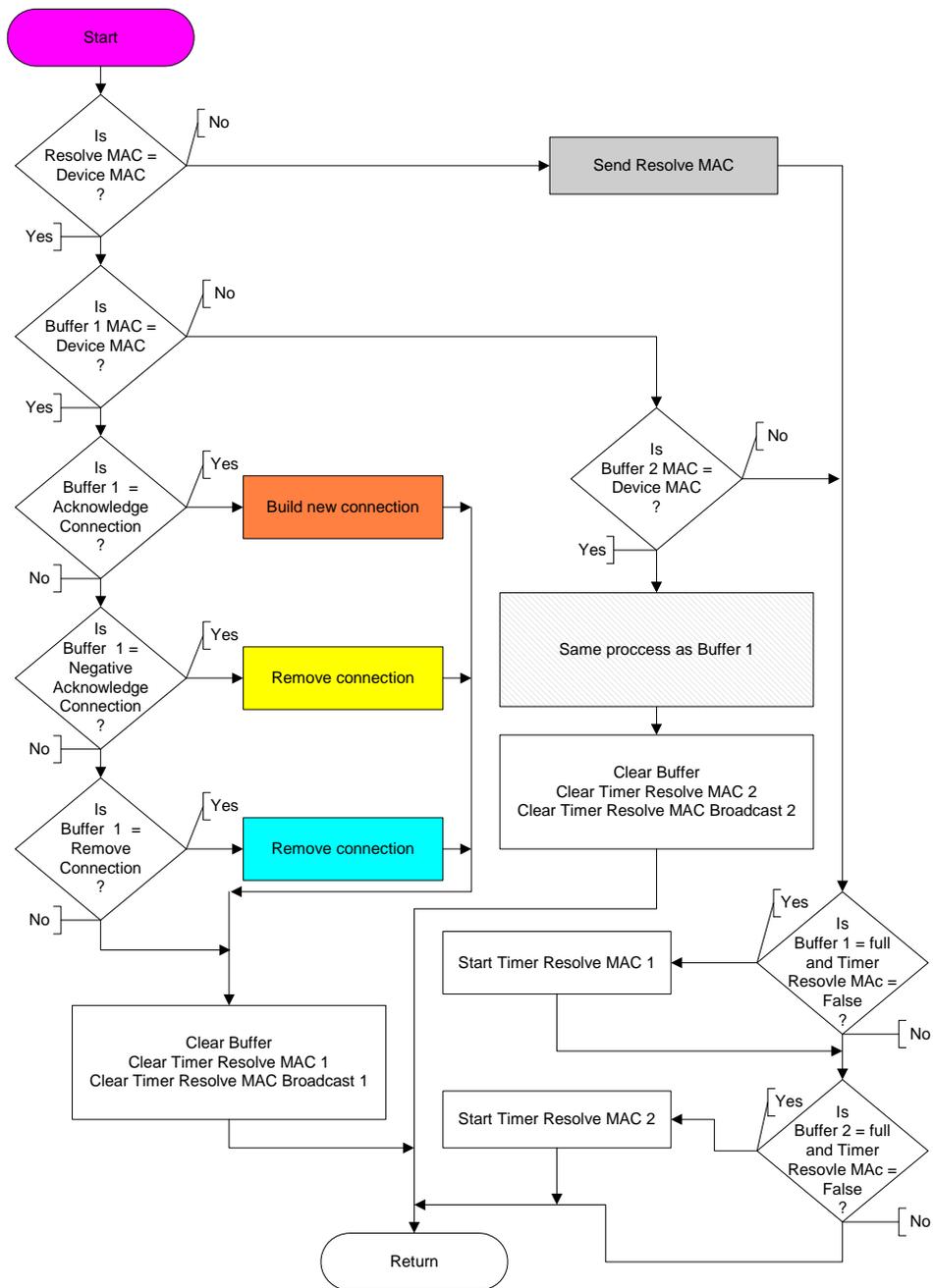


Figure 5.8 Flow diagram for Resolve MAC address request

Secondly, the device needs to buffer the Establish connection and endpoint status command before a MAC address resolve command is sent. Each device will have two buffers. Each buffer must be able

to record the remote device MAC address, short address and command to send after a MAC address is resolved.

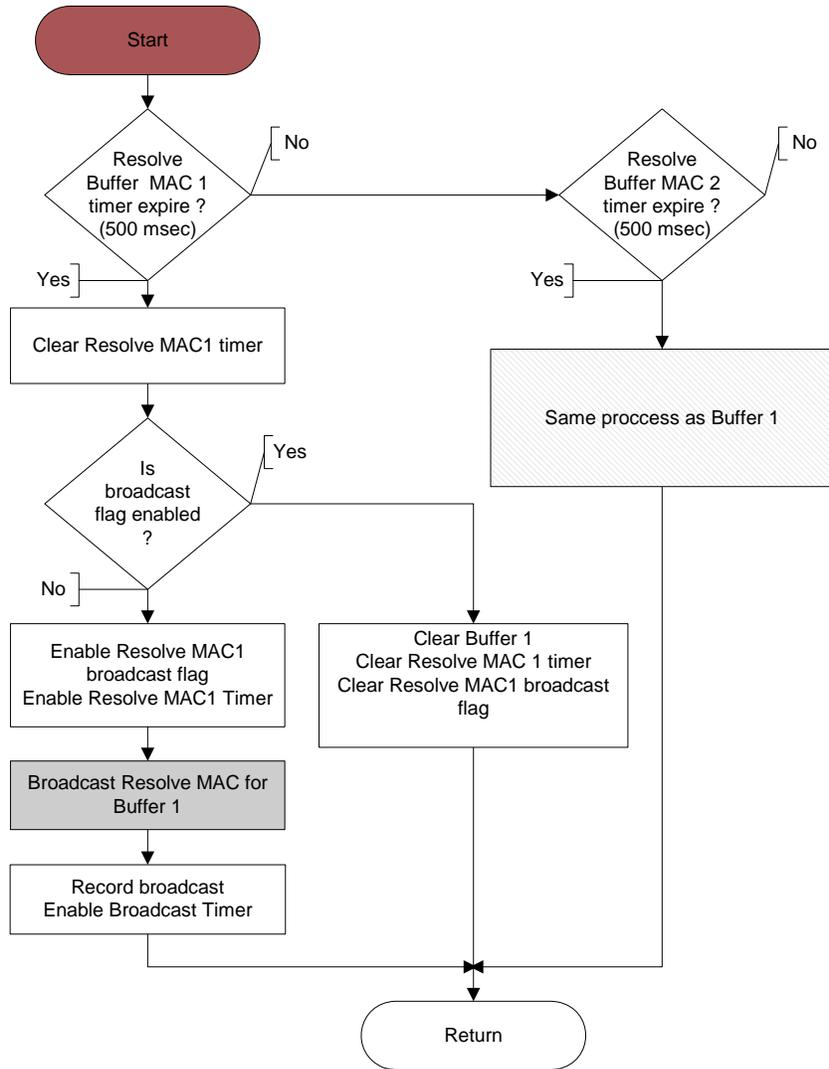


Figure 5.9 Flow diagram for MAC address timer

Figure 5.8 illustrates the flow diagram for the Resolve MAC address request. The device must first investigate if the requested MAC address does not belong to the device before the Resolve MAC address request is sent. If the buffered MAC address belongs to the device, the request will be

processed. Only the buffer with the device MAC address will be processed. Lastly the buffer and timers associated with the buffer is cleared.

If the Resolve MAC address request is sent, a Resolve MAC address timer is started. If the timer expires, the MAC address is broadcasted to all devices. As illustrated in Figure 5.9 there is a timer for each buffer. If a Resolve MAC x timer expires and a broadcast was not sent previously a broadcast flag is enabled and the Request is retransmitted as a broadcast. The broadcast is recorded and the device will not forward the recorded broadcast for a pre-defined time. (This will be discussed in the section about how frames are forwarded in section 5.2.1.) If the Resolve MAC address timer expires again, the buffer is cleared. The endpoint status request function will initialise another Resolve MAC address request every 60 to 120 seconds, depending on the IO type and this will repeat the Resolve MAC address procedure.

Figure 5.10 illustrates the three types of MAC address commands that a device can receive. They are:

- *Resolve MAC address request*

When a device receives a unicast Resolve MAC address request and the requested short address matches the MAC address of the device or receives a broadcast Resolve MAC address request, the device will then update its entire Remote endpoint table. For each entry in the table where the Remote MAC address matches the MAC address of the requester the Remote short address is updated. (This table will be discussed in the section about management of connections function development in section 5.4.) If the MAC address does not match a unicast request, a Negative acknowledge MAC address is returned. However, if the MAC address matches a unicast or broadcast MAC address request, an Acknowledge MAC address is returned to the requester.

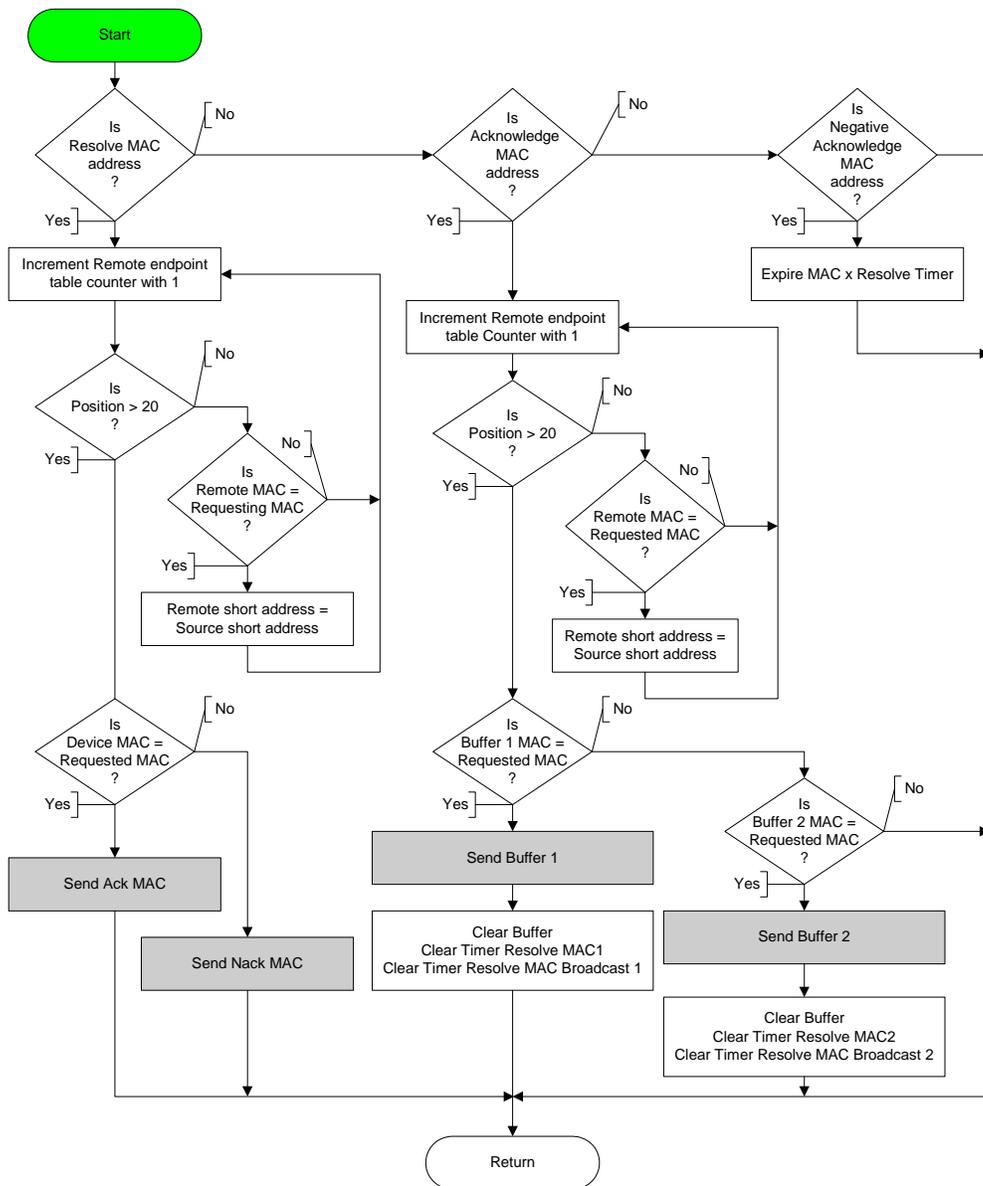


Figure 5.10 Flow diagram for MAC address data received

- Acknowledge MAC address

When a device receives an Acknowledge MAC address, it will first update its Remote endpoint table. For each entry where the MAC address field matches the Acknowledge MAC address, the short address field is updated with the received start source short address field of

the frame. Secondly the MAC address is compared with both buffered MAC addresses. The contents of the matching buffers are transmitted and the contents and timers associated with the buffer are cleared.

- *Negative acknowledge MAC address*

A device will only receive a Negative acknowledge MAC address for a unicast Resolve MAC address request. When the above is received the device will set the Resolve MAC address timer of the associated buffer to expire. This will initiate the device to broadcast a Resolve MAC address request as discussed in Figure 5.9.

## 5.4 Management of connections function development

### 5.4.1 Connection tables

In the section about establishing connections between endpoints, the functionality of the Group, Capability and IO of the endpoint was discussed (section 3.2.3.2). Table 5.1 to Table 5.3 illustrates the values that were used when the protocol was designed. The ID will be hard coded into the device and the manufacturer can decide what the values for each field must be.

*Table 5.1 Endpoint groups*

Group ID	Description
1	Security
2	Air-conditioning
3	Lighting
4	Irrigation
..255	Spare

*Table 5.2 Endpoint capabilities*

Capability ID	Description
1	Digital
2	Analogue
..16	Spare

Table 5.3 Endpoint IO's

IO ID	Description
1	Input
2	Output – Single
3	Output – And
4	Output – Or
..16	Spare

A Remote endpoint table is required for a device to setup, maintain and remove a connection and is illustrated in Table 5.4.

Table 5.4 Device remote endpoint table

Local ID/2	Keep-alive Timer	MAC	Remote ID	Capability	IO	Short Address	Description	Status	Send Flag
1									
:									
20									

When a connection to another endpoint is established, the local devices must have a local record of the remote devices MAC and short addresses. The MAC address is required when the remote devices short address changes and the local device needs to discover the new short address of the remote device. The remote ID is used to address the correct endpoint of the device. If it is an “AND” or “OR” IO type, the protocol allows for an output endpoint to link two inputs..This is the reason that the remote endpoint table has double the entries than the local endpoints. For example entry 5 and 6 will be the remote endpoints for endpoint 3.

The Keepalive timer, Description, Status and Send Flag fields are only used after the connection is established and will be discussed in the section about managing endpoint statuses in section 5.5.

Table 5.5 Coordinator request connection table

Entry	Keep-alive Timer	End-point ID	Group	Capability	IO	MAC Address	Key	Short Address
1								
:								
10								

The simulator design allows the coordinator to setup ten simultaneous connections. If the coordinator does not find a partner for the first request, a timer will expire and the coordinator will clear the request. The coordinator records the requesting endpoints group, capability and IO and this allows the coordinator to investigate the second request. If the first and second request matches, an acknowledgement is sent to both requesting devices, otherwise a Negative acknowledgement is sent. The MAC and short addresses are recorded and this allows the coordinator to confirm the short addresses of the requesting devices before a reply is sent. For the coordinator to match two requests a unique key is used with every connection request. The coordinators request connection table is shown in Table 5.5.

The algorithm allows a device to be the coordinator, local and remote device on the same device or on separate devices. If the device changes from a coordinator to a parent, the current Request connection table is deleted without any replies to the requesting devices. The protocol uses timers to clear the requesting devices outstanding requests. The devices will resubmit their request to the new coordinator.

### **5.4.2 Frame format**

The Manage connection suite consists of four commands. The frame format of the commands is almost the same, but the field value differs for each command. Bit 4 of the Connection control field, the Connection bit, indicates all manage connection commands.

- *Request connection*

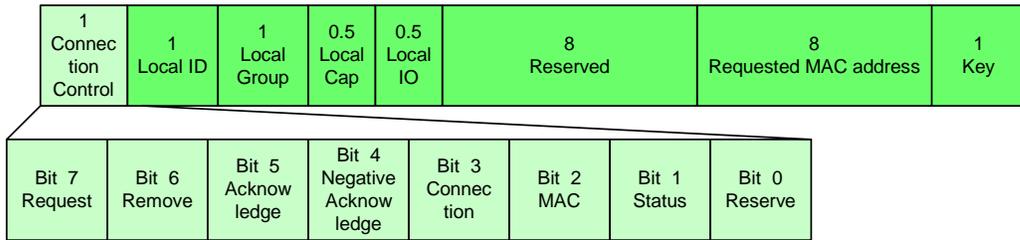


Figure 5.11 Request connection frame

Figure 5.11 illustrates the Request connection frame format. The number above each field indicates the length in bytes of each field.

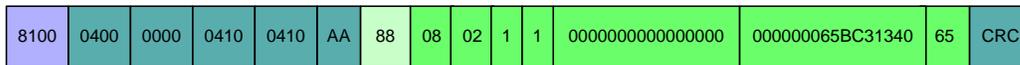


Figure 5.12 Request connection frame sample

A device will send a Request connection command to the coordinator when a new connection between two endpoints is requested. Even when both endpoints are located on the same device, a request for each endpoint is sent to the coordinator. Figure 5.12 illustrates a device with a short address 0x0410 and a MAC address 0x000000065BC31340 sending a Request connection to the coordinator via device 0x400. This endpoint is a light switch and is defined in the frame as follows; ID is 0x08, belongs to the lighting group and is a digital input. The coordinator will use the key 0x65 to find a matching endpoint.

- Acknowledge connection

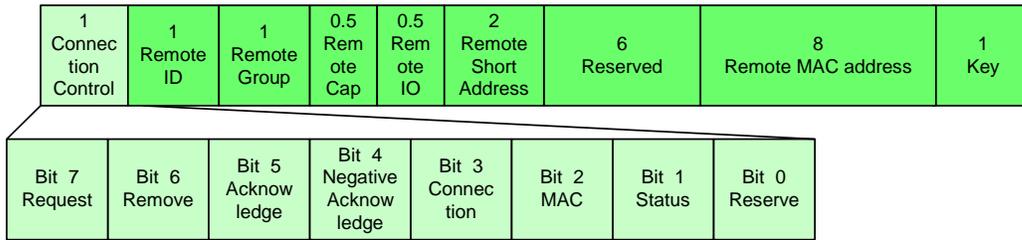


Figure 5.13 Acknowledge connection frame

When the coordinator finds a match for a request, an Acknowledge connection response is sent to both requesting devices. Figure 5.13 illustrates the frame format for the Acknowledge connection command. It must be highlighted that this frame represents the characteristics of the remote endpoints.

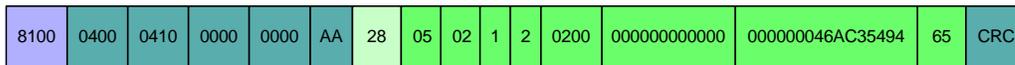


Figure 5.14 Acknowledge connection frame sample

Figure 5.14 illustrates the Acknowledge connection response for the Request connection sent in Figure 5.12. The MAC address of the remote device is 0x000000046AC35494 and its short address is 0x0200. The remote endpoint ID is 0x05, belongs to the lighting group and is a digital output. This is the characteristic of a light bulb. Device 0x0200 will also receive an Acknowledge response, but with device 0x410 characteristic. Both devices store the connection information in their Remote endpoint tables. The table must be stored in a non-volatile memory to allow the device to retain the connections after a power failure.

- Negative acknowledge connection

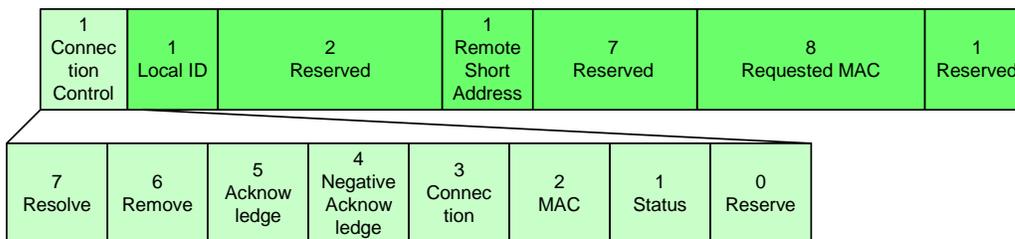


Figure 5.15 Negative acknowledge frame

Figure 5.15 illustrates the Negative acknowledge connection frame format. The coordinator only needs to specify the local endpoint ID to the requester.

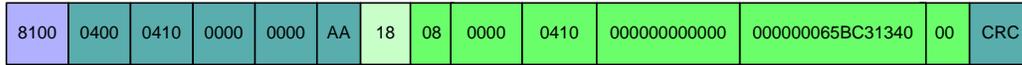


Figure 5.16 Negative acknowledge connection frame sample

Figure 5.16 illustrates the Negative acknowledge connection response for the Request connection in Figure 5.12. The coordinator will send a Negative acknowledge connection response if it does not find a matching request in a pre-defined period or when the capabilities of the two matching requests do not match.

- *Remove connection*

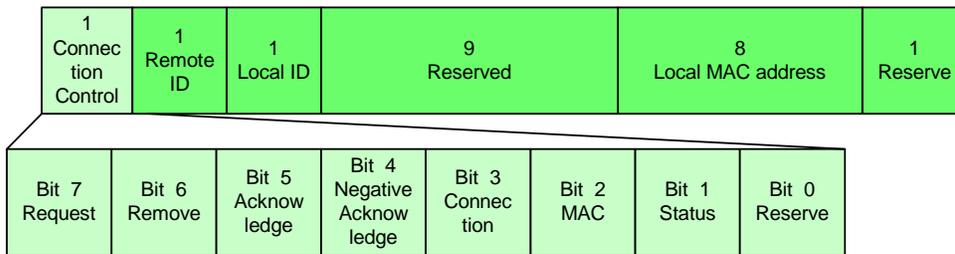


Figure 5.17 Remove connection frame

If a device requires removing a connection it will send a Remove connection request to the remote device and remove the connection from its Remote endpoint table. Figure 5.17 illustrates the frame format of the Remove connection command. The command only specifies the Local ID, Remote ID and the local device MAC address. This, including the short address of the device, is the only information necessary for the remote device to remove the connection from its Remote endpoint table.

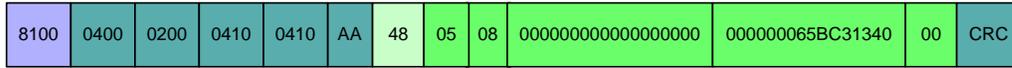


Figure 5.18 Remove connection frame sample

Figure 5.18 illustrates when device 0x000000065BC31340, with short address 0x0410 sends a Remove connection request. It will request device 0x0200 to delete endpoint 0x05 connection to device's endpoint 0x08.

### 5.4.3 Algorithms

Figure 5.19 illustrates the flow diagram when a device requests the setup of a connection between two endpoints. This device will first validate that it has a valid short address.

The protocol allows the device to setup two connections simultaneously. It must be able to setup a connection between two endpoints hosted on the same device and therefore it must first inspect which Request connection buffer is empty. The device will drop the request if both buffers are occupied otherwise it will record the endpoint's ID, group, capabilities IO and key.

The associated Request connection timer is enabled. If the device is the coordinator the request is recorded in the Request connection table of the coordinator otherwise the request is sent to the coordinator.

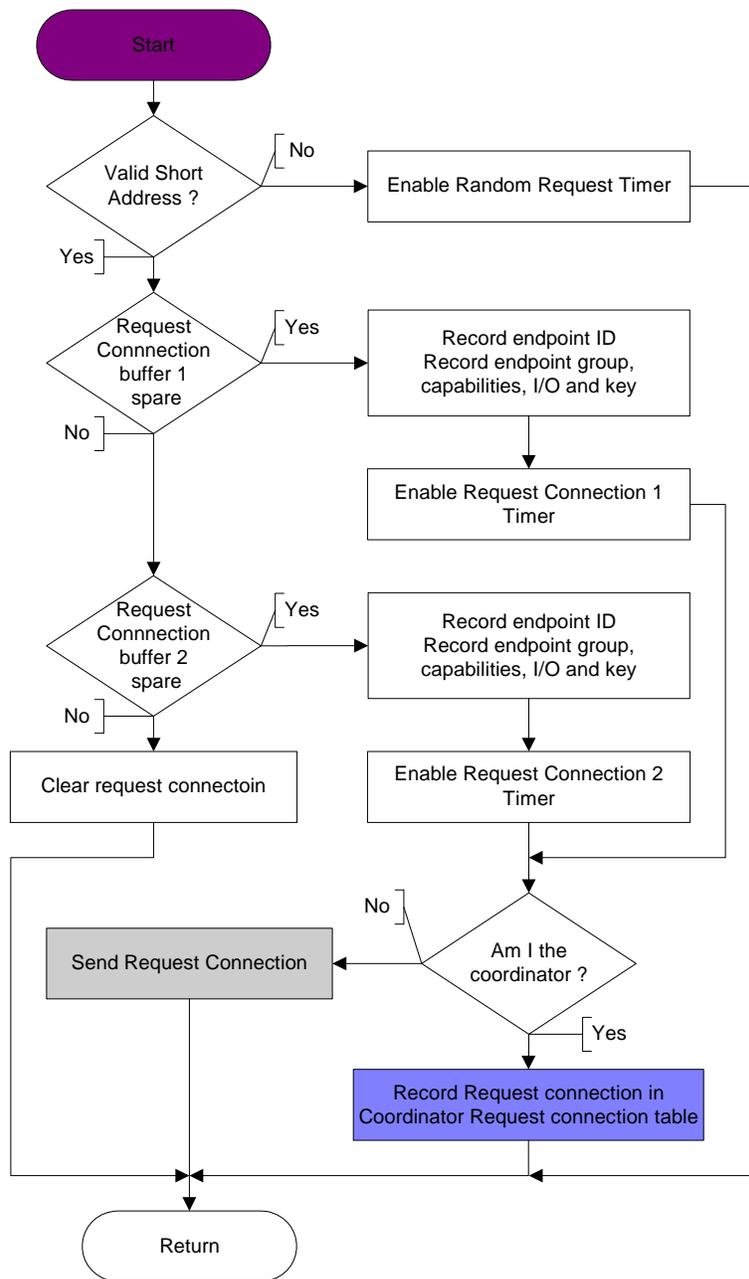


Figure 5.19 Flow diagram for Request connection send

Figure 5.20 illustrates the flow diagram when the coordinator receives a Request connection frame. If the requester is the coordinator, the request can be received via the transmission media or internally. The coordinator will first inspect all ten entries in the Request connection table for a matching key.

The partner found flag is set and both request capabilities are compared. If both requests match, two Acknowledge connection replies are stored in the transmit buffer. The first is for the previously recorded request and the second buffer is for the last received request. The Acknowledge connection reply contains all the endpoint information of the remote device. This will enable the device to learn the connection information of the remote device. If the above request does not match, both buffers are loaded with Negative acknowledge connection replies. The MAC and short address of both devices are recorded before the Request connection table entry is cleared. If both requests came from the same device, only one Resolve MAC address request is sent, if not, a request is sent to both devices. The command is transmitted after the MAC address is resolved. This was discussed in the section about the MAC address command in section 5.3.

If no matching key is found the new request is recorded in the first empty entry of the Request connection table. A Request connection coordinator timer is started. If this timer expires, the request is terminated and a Negative Acknowledge connection reply is sent to the requester. This will be discussed later in the section about Connection establish timers.

Figure 5.21 illustrates the flow diagram when a device receives an Acknowledge connection reply. If the device is the coordinator, the command can be received via the transmission media or internally.

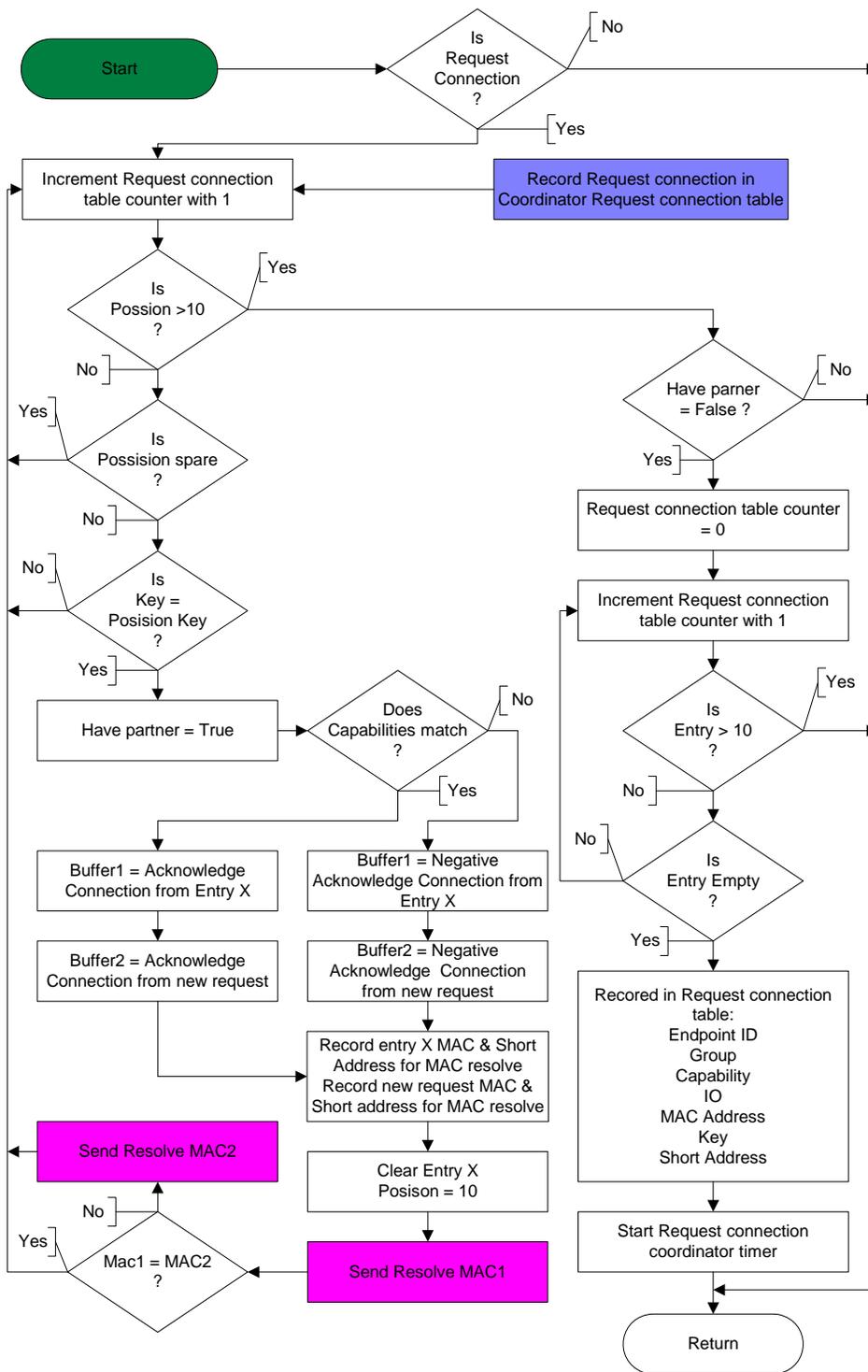


Figure 5.20 Flow diagram for Request connection data received for the coordinator



The Acknowledge connection reply only informs the device about the endpoint information, including the key of the remote device. If the remote endpoint is hosted on another device, the device only needs to inspect which Request connection buffer has the same key as in the Acknowledge connection command. Each local endpoint can have a connection to two endpoints depending on the I/O capabilities of the endpoint. The Remote endpoint table entry for an endpoint is calculated as follows:

$$\text{First Entry} = \text{Local ID} \times 2 - 1. \quad (1)$$

$$\text{Second Entry} = \text{Local ID} \times 2. \quad (2)$$

This was explained in the section about Connection tables in section 5.4.1. The connection will be recorded in the first spare Remote endpoint table entry for that specific local endpoint. The ID, Capabilities, I/O of the remote endpoint and the MAC and short address of the device are recorded in the Remote endpoint table. The Keepalive value is set to 118. This value will increase every second. When the value is greater than 120, the device will send a request to the remote device, requesting for an update on the status of the remote device. This will be discussed in the next section about managing endpoint statuses in section 5.5. Lastly the Request connection buffer and the Request connection timer are cleared.

Both endpoints are hosted on the same device, when a device receives an acknowledgement and the Remote MAC address is the same as the MAC address of the device. The device cannot link the first Request connection buffer with the key in the same way as when the endpoints are hosted on different devices. The problem is that both Connection buffers' key are the same and the device is going to receive two acknowledgements, one for each side of the connection. In this case the device will inspect the Local ID of the Connection buffer and link the buffer to the acknowledgement with a different ID. The information of the remote endpoint is recorded in the same way as when the endpoints are hosted on separate devices.

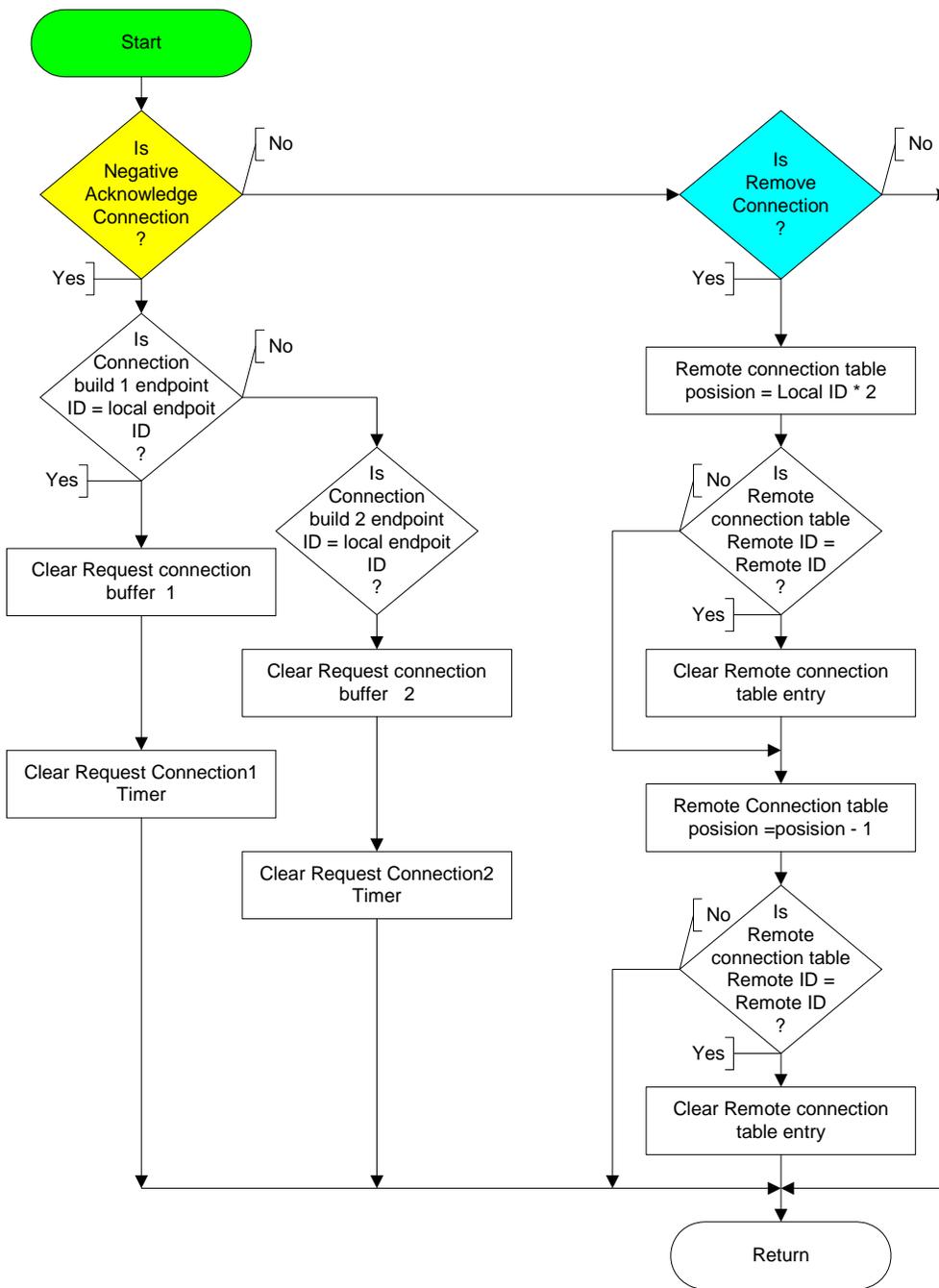


Figure 5.22 Flow diagram for Negative acknowledge and Remove connection request

Figure 5.22 illustrates the flow diagram when the device receives a Negative acknowledge connection reply or a Remove connection request via transmission or internally. A Negative acknowledge

connection is processed internally when the coordinator hosts the endpoint. A Remove connection is processed internally when both endpoints are hosted on the same device.

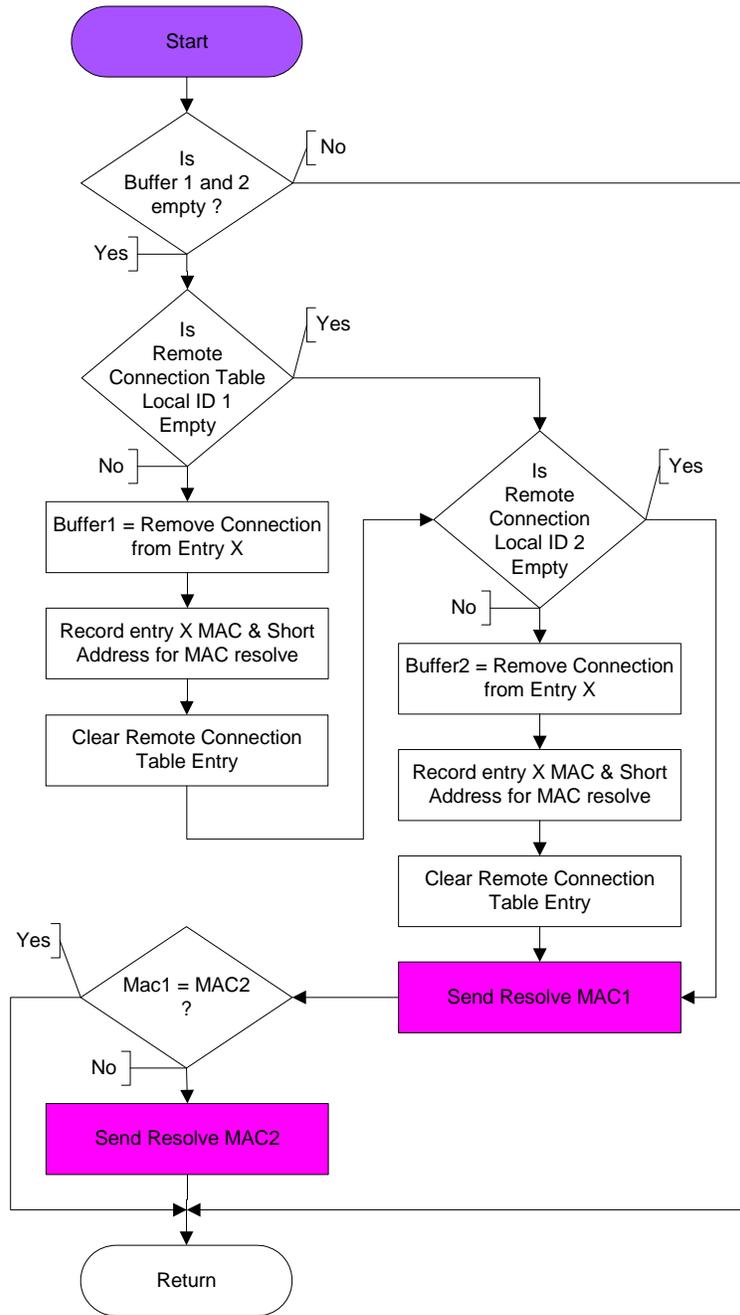


Figure 5.23 Flow diagram for Remove connection request

When a Negative acknowledge connection reply is received, both Connection buffers are inspected and cleared. This happens where the recorded ID matches the ID specified in the Negative acknowledge connection reply and the associated Request connection timer is cleared.

The device must inspect both Remote endpoint table entries for the Local ID specified in the Remove connection request. The connection is cleared if the remote ID matches the remote ID specified in the Remove connection request.

Figure 5.23 illustrates the flow diagram when a device requests to remove a connection. Some endpoints can have two remote endpoints and this is why both buffers must be empty. The first buffer is used for the first connection and the second buffer for the second connection.

The associated Remove connection request is loaded into the buffer. The MAC and short address are recorded to confirm the short address of the device with a Resolve MAC address request after the Remote endpoint table is cleared for the specific Local ID. Only one Resolve MAC address request will be sent if both remote endpoints are hosted on the same device.

Figure 5.24 illustrates the flow diagram for the Connection establish timer procedures. This procedure is part of all the previous timer procedures discussed. There are two types of Connection establish timers; one is for the requester and the other for the coordinator Request connection.

A timer is started when a device requests a connection for an endpoint. A device can request two simultaneous connections and will start a Request connection timer for each. The Request connection timer was started for each Request connection illustrated in Figure 5.19. If no reply is received from the coordinator for a request, the associated Request connection buffer is cleared when the associated Request connection timer expires.

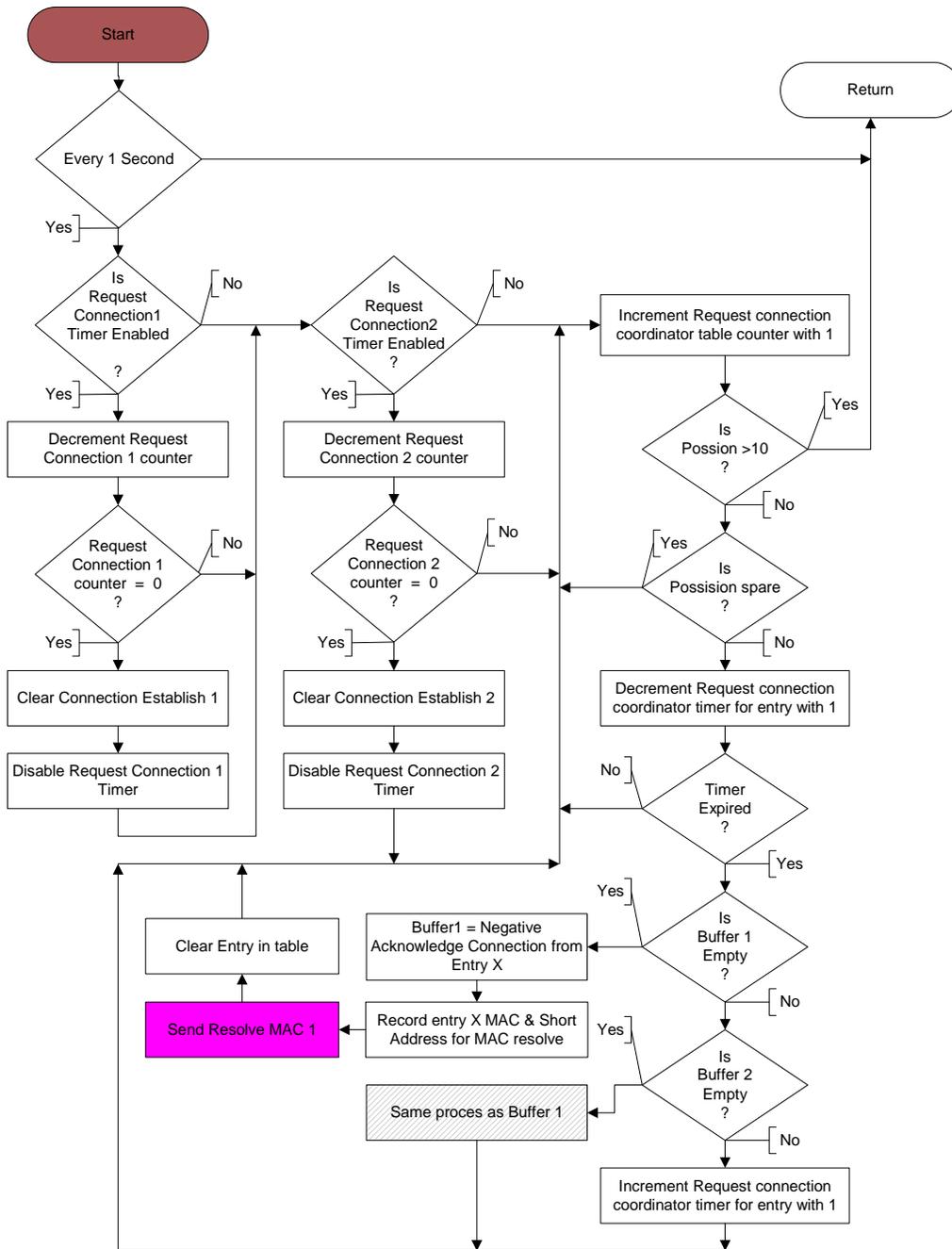


Figure 5.24 Flow diagram for connection establish timers

When the coordinator receives a Request connection, it is stored in the Request connection coordinator table and a counter is started that is associated with that entry. All entries in the table are

inspected every second and if an expired entry is found, a Negative acknowledge connection reply is sent to the requester.

Before the command is sent, the short address must be confirmed and the command must be stored in one of the two buffers. If both buffers are occupied, a second is added to the timer so that it will expire again in the next second. If an empty buffer is found, the MAC resolve process is started. (This was discussed in the section about the Resolve MAC address in section 5.3.2.) Lastly the entry in the Request connection coordinator table is cleared.

## 5.5 Management of endpoint status function development

When a connection is established between two endpoints, both endpoints must be updated. The endpoints will update each other periodically or when the status of an input endpoint changes.

### 5.5.1 Frame format

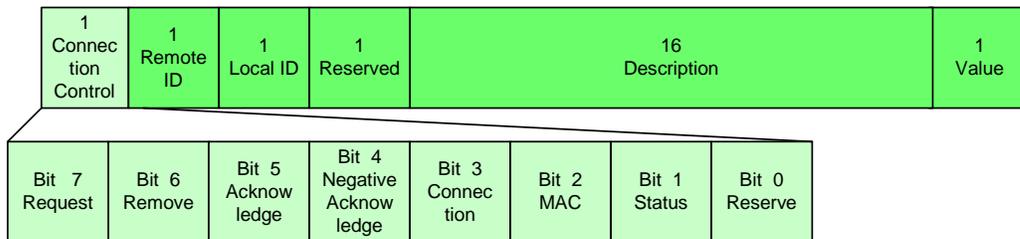


Figure 5.25 Request endpoint status frame

The Payload of the frame in Figure 5.1, changes as illustrated in Figure 5.25. The Manage endpoint status suite consists of two commands and bit 1 of the Connection control field indicates any Manage endpoint status suite commands. The commands will always include the description and status of the local endpoint. The commands are:

- *Request endpoint status*

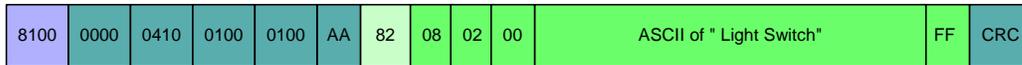


Figure 5.26 Request endpoint status frame sample

Figure 5.26 illustrates an example frame of a Request endpoint status command. In the above frame, device 0x100, requests device 0x0410 endpoint 0x8 status. The local endpoint ID is 0x2. The description of the local endpoint is ‘Light switch’ and its status is 0xFF.

- *Acknowledge endpoint status*

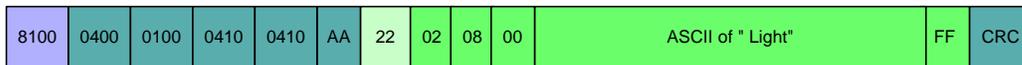


Figure 5.27 Acknowledge endpoint status frame sample

Figure 5.27 is an Acknowledge endpoint status reply example for the request sent in Figure 5.26. Device 0x0410 replies to device 0x100 that endpoint 0x8 status is 0xFF and its description is ‘Light’.

An important design criteria is that an output IO can be a ‘AND’ or an ‘OR’, output and that it is then possible for an output endpoint to update the status of an input endpoint. An example is when an ‘OR’ output has a connection to two input endpoints with the status of both input endpoint off. The status of input #1 changes to on, and it sends a status update to the output endpoint. The output device calculates its output and sets the endpoint. The device sends an Acknowledge endpoint status to the requester, input #1, but also sends a Request endpoint status to the endpoint of input #2.

## 5.5.2 Algorithms

Figure 5.28 illustrates the flow diagram for a Request endpoint status transmission. When an endpoint needs to request the status of its remote endpoint, the hosting device must first determine if the local and remote endpoints are hosted on the same device. If the above is true, the device will record the Remote ID, Local short address, Description and command type and send it directly to the procedure that would receive the command as illustrated in Figure 5.28. The command type depends on the fact if the command is a Request or Acknowledgement endpoint.

If the command must be transmitted to another device, the device must determine if the command must be a Request endpoint status or Acknowledge endpoint status. The device will store the command in the first available Connection establish buffer. If no buffer is available the command is ignored, otherwise the device will resolve the MAC address of the device as discussed in the section about Resolve MAC address request in section 5.3.2.

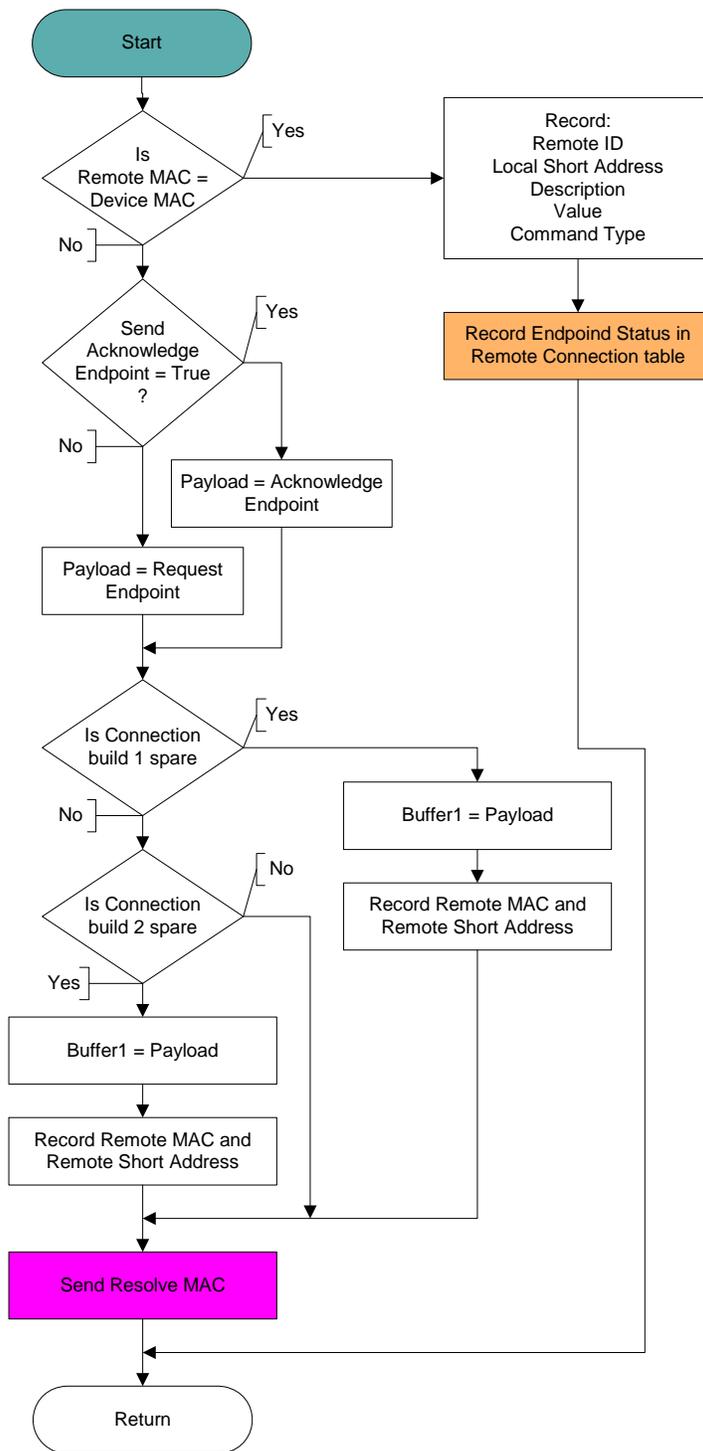


Figure 5.28 Flow diagram for Request endpoint status transmitted

A request can be sent when the status of an endpoint changes or with periodic updates. Figure 5.29 illustrates the two flow diagrams associated with the endpoint status process.

Figure 5.29 (a) illustrates the process where the device inspects each entry of the Remote endpoint table. The device will send a Request endpoint status command, if the timer is more than 120 seconds and if a pending Request endpoint status command is not outstanding. In the section about when a connection is established under Figure 5.21 the counter is set to 118 seconds when a connection is established.

Two seconds after a connection was established, a status update is requested for the linked endpoint and a flag is set in the Remote endpoint table that a request was sent. If the counter reaches 130 the indication of the remote endpoint is disabled. If the counter reaches 140 the flag is cleared and the counter is set to 130. This will initialise another Request endpoint status command every ten seconds.

Figure 5.29 (b) illustrates the process of how the device increases the Keepalive endpoint counter for each active entry in the Remote endpoint table.

Figure 5.30 illustrates the flow diagram for a device receiving a Request endpoint status frame. The only difference between a Request endpoint status and Acknowledge endpoint status frame structure is established in their Connection control field. The functionality only differs in the fact that an acknowledgement is a reply on a request and nothing is returned when an acknowledgement is sent.

When a Management endpoint frame is received the device investigates whether it is a request or an acknowledgement frame. If the command received is a request, a Send acknowledgement flag is set, indicating that a reply must be sent. From here the process for a command received via the transmission medium happens in the same way as if it was received from an internal procedure.

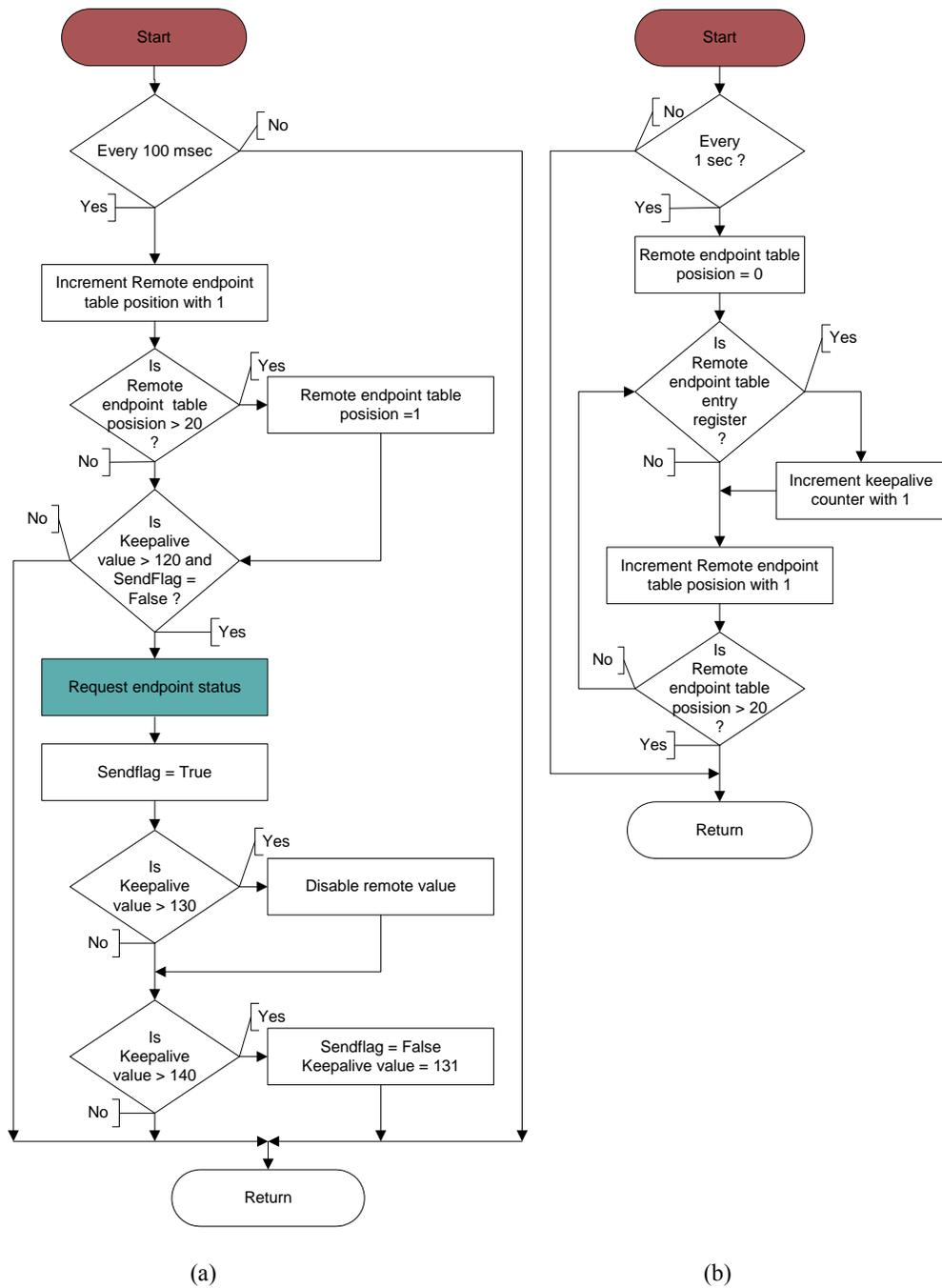


Figure 5.29 Flow diagram for endpoint status timers

First in the Remote endpoint table the entry 's position is investigated:

$$\text{Entry} = \text{Local ID} \times 2 - 1. \quad (3)$$

If the requested short address and remote ID in the Remote endpoint table is the same as the received frames, the remote description and value are recorded in the Remote endpoint table. The Send flag is cleared and if this is an acknowledgement for a pending status request, the Connection Keepalive counter is also set to 60 seconds. If the endpoint is an output, the Connection Keepalive counter is set to one; this is done to keep an output endpoint from sending a Request endpoint status under normal conditions. An input endpoint Connection Keepalive counter will always expire 60 seconds before an output endpoint Connection Keepalive counter. The output endpoint will request a status update only when the connection is temporarily broken and the output endpoint does not receive any updates from its input endpoints. The device will update the remote endpoint indication and calculate the local endpoint value if it is an output endpoint.

If the Send acknowledgement flag is set for a local endpoint, an Acknowledge endpoint status frame is sent for each Remote endpoint. But in section 5.5.1 it was mentioned that the output must send an acknowledgement back to the requester and send a Request endpoint status to the other device to update the other device about the status change. To save bandwidth and shorten the process, the following points were taken into account:

- Only an output can have two remote endpoints.
- If an input changes its status, a Request endpoint status frame is sent immediately.
- When an output endpoint changes its status, only the remote device indication for the output endpoint needs to be updated.



Lets investigate the following scenario:

- Input #1 = Off
- Input #2 = Off
- Output (And) = Off

The status of Input #1 changes to 'On' and sends a Request endpoint status to the output endpoint. The output endpoint updates its indication for Input #1 to 'On' and changes the output status to 'On'. An acknowledgement is send to Input #1 and #2. Input #1 clears its pending flag and updates its indication to 'On' for the remote endpoint. Input #2 receives an acknowledgement and does not clear the pending flag. It updates its indication to 'On' for the remote endpoint. All local and remote indications are now updated with one less acknowledgement sent. If the status update of Input #2 was lost, it will get updated with the normal endpoint Keepalive function as discussed earlier.

The same process is repeated for the second Remote endpoint where the entry is:

$$\text{Entry} = \text{Local ID} \times 2.$$

## 5.6 Simulation and results

The simulate tab was used to simulate the endpoints, as discussed in section 4.2.4, and the trace tab, that was discussed in section 4.2.3, was used to capture all raw data. A series of simulations was done to prove the functionality of the algorithms described in this chapter.

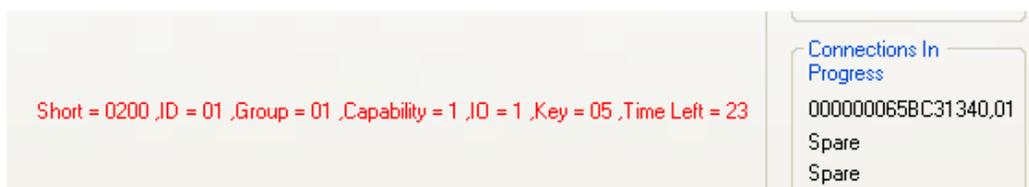


Figure 5.31 Simulator snapshot of coordinator

Figure 5.31 illustrates a snapshot of the simulator tab of the coordinator while a connection is established. The complete screen is shown in Figure 4.4. The Connection in progress section displays the MAC address and Local ID of the endpoint and reveals the information inside the coordinator request connection table. Because of space limitations only the two fields are displayed, but if the mouse is moved over the entry all the other fields of the table are displayed in red on the left-hand side.



Figure 5.32 Simulator snapshot of a connection establish between to endpoints

Figure 5.32 illustrates a snapshot of the simulator tab when a connection is established between two endpoints and the following information can be read: The Local ID is 0x01 and the input is in an ‘ON’ position. The MAC address of the remote endpoint’s hosting device is 0x0000000874B38C2E and it has a short address of 0x0200. The ID is 0x02 and the Keepalive timer is 77.

When a connection is established, the user must tick the select button of an endpoint, enter a key, ‘5’ (used in above example) and select the Link button. When an input value needs to change the user must tick the select button of an endpoint, enter a value and select the Change value button. For an analogue input the value can range from 0 to 255 and for a digital input the value can be 0 or 255. If any value between 1 and 255 is entered, the value will be converted to 255. When the user needs to remove a connection it must tick the selection button of the endpoint and then select the Delete

button. By default only the buttons that are valid are enabled. If a single output endpoint already has a connection, only the Reset, Change value and Delete buttons are enabled, but if the output type was an ‘AND’ output, the Link button would also be enabled. The reset button is used if a selection needs to be cancelled.

### 5.6.1 Unsuccessful connection established between two endpoints

In the first simulation three devices are setup to be in the same broadcast domain. Device 1 is the coordinator, Device 2 is the light switch and Device 3 the siren. The packet captures of Device 1 to 3 is shown from Table 5.6 to Figure 5.8. Capturing started after the devices received their short addresses. All transmitted packets are indicated in blue and received in red.

Table 5.6 Unsuccessful connection. Device 1: MAC 0000001641A75694

Step	Device 1: MAC 0x0000001641A75694, Short 0x0000
1	8100000000002000200AA88010111000000000000000000000874B38C2E05C031
2	8100000000001000100AA8803031100000000000000000000065BC31340058C79
3	8100020002000000000AA840000000000001641A756940000000874B38C2E003F56
4	8100010001000000000AA840000000000001641A75694000000065BC3134000C4CA
5	8100000000002000200AA24000000000000000000000000000874B38C2E00819E
6	8100020002000000000AA180100002000000000000000000000874B38C2E0500CD
7	8100000000001000100AA2400000000000000000000000000065BC31340001C51
8	8100010001000000000AA18030000100000000000000000000065BC3134005E837

Table 5.7 Unsuccessful connection. Device 2: MAC 000000065BC31340

Step	Device 2: MAC 0x000000065BC31340, Short 0x0100
1	
2	8100000000001000100AA8803031100000000000000000000065BC31340058C79
3	
4	8100010001000000000AA840000000000001641A75694000000065BC3134000C4CA
5-6	
7	8100000000001000100AA2400000000000000000000000000065BC31340001C51
8	8100010001000000000AA18030000100000000000000000000065BC3134005E837

Table 5.8 Unsuccessful connection. Device 3: MAC 0000000874B38C2E

Step	Device 3: MAC 0x0000000874B38C2E, Short 0x0200
1	8100000000002000200AA88010111000000000000000000000874B38C2E05C031
2	
3	81000200020000000000AA84000000000001641A756940000000874B38C2E003F56
4	
5	8100000000002000200AA24000000000000000000000000000874B38C2E00819E
6	81000200020000000000AA18010000020000000000000000000000874B38C2E0500CD
7-8	

The following happened at each step in Table 5.6 to Figure 5.8:

1. Device 3, with short address 0x0200 and MAC address of 0x0000000874B38C2E, sends a Request connection to the coordinator. The ID of the endpoint is x0x01, it belongs to the security group, has a digital capability and is an input. The request key is 0x05.
2. Device 2, with short address 0x0100 and MAC address of 0x000000065BC31340, sends a Request connection to the coordinator. The ID of the endpoint is 0x03, it belongs to the lighting group, has a digital capability and is an input. The request key is 0x05. The coordinator compares both requests and finds that their capabilities do not match.
3. The coordinator sends a Resolve MAC request to device 0x0200 confirming that its MAC address is still 0x0000000874B38C2E.
4. The coordinator sends a Resolve MAC request to device 0x0100 confirming that its MAC address is still 0x000000065BC31340.
5. Device 3 sends an Acknowledge MAC frame to the coordinator.
6. Device 1, the coordinator, sends a Negative acknowledge connection reply to Device 3.
7. Device 2 sends an Acknowledge MAC frame to the coordinator.
8. Device 1 sends a Negative acknowledge connection reply to Device 2.

### 5.6.2 Successful connection established between two endpoints

In this simulation three devices are setup to be in the same broadcast domain. Device 1 is the coordinator, Device 2 is the door sensor and Device 3 the siren. The packet captures of Device 1 to 3

are shown from Table 5.9 to Table 5.11. Capturing started after the devices received their short addresses. All transmitted packets are indicated in blue and received in red.

Table 5.9 Successful connection. Device 1: MAC 0000001641A75694

Step	Device 1: MAC 0x0000001641A75694, Short 0x0000
1	8100000000001000100AA8801011100000000000000000000065BC31340055DFE
2	8100000000002000200AA880201130000000000000000000000874B38C2E05802A
3	8100010001000000000AA84000000000001641A75694000000065BC3134000C4CA
4	8100020002000000000AA84000000000001641A756940000000874B38C2E003F56
5	8100000000002000200AA240000000000000000000000000000874B38C2E00819E
6	8100020002000000000AA280101110100000000000000000000065BC31340055A61
7	8100000000001000100AA240000000000000000000000000000065BC31340001C51
8	8100010001000000000AA2802011302000000000000000000000874B38C2E0502C3
9	8100000020001000100AA8400000000000065BC31340000000874B38C2E00BB4A
10	7100020002000000100AA840000000000065BC31340000000874B38C2E00CFD6
11	81000000010002000200AA240000000000000000000000000000874B38C2E002DFE
12	7100010001000000200AA240000000000000000000000000000874B38C2E006DDA
13	81000000020001000100AA82020100446F6F722053656E736F7220202020FF8F0D
14	7100020002000000100AA82020100446F6F722053656E736F7220202020FFFB91
15	81000000010002000200AA84000000000000874B38C2E00000065BC31340004BE8
16	7100010001000000200AA84000000000000874B38C2E00000065BC31340000BCD
17	81000000020001000100AA2400000000000000000000000000065BC313400054B2
18	7100020002000000100AA2400000000000000000000000000065BC3134000202E
19	81000000010002000200AA22010200536972656E20202020202020202020FFED06
20	7100010001000000200AA22010200536972656E20202020202020202020FFAD23

Table 5.10 Successful connection. Device 2: MAC 000000065BC31340

Step	Device 2: MAC 0x000000065BC31340, Short 0x0100
1	8100000000001000100AA8801011100000000000000000000065BC31340055DFE
2	
3	8100010001000000000AA84000000000001641A75694000000065BC3134000C4CA
4-5	
7	8100000000001000100AA240000000000000000000000000000065BC31340001C51
8	8100010001000000000AA280201130200000000000000000000874B38C2E0502C3
9	8100000020001000100AA8400000000000065BC31340000000874B38C2E00BB4A
10	
12	7100010001000000200AA240000000000000000000000000000874B38C2E006DDA
13	8100000020001000100AA82020100446F6F722053656E736F7220202020FF8F0D
14-15	
16	7100010001000000200AA84000000000000874B38C2E00000065BC31340000BCD
17	8100000020001000100AA2400000000000000000000000000065BC313400054B2
18-19	
20	7100010001000000200AA22010200536972656E20202020202020202020FFAD23

Table 5.11 Successful connection. Device 3: MAC 0000000874B38C2E

Step	Device 3: MAC 0x0000000874B38C2E, Short 0x0200
1	
2	8100000000002000200AA88020113000000000000000000000874B38C2E05802A
3	
4	8100020002000000000AA8400000000001641A756940000000874B38C2E003F56
5	8100000000002000200AA24000000000000000000000000000874B38C2E00819E
6	8100020002000000000AA2801011101000000000000000000065BC31340055A61
7 - 9	
10	7100020002000000100AA840000000000065BC31340000000874B38C2E00CFD6
11	81000000010002000200AA240000000000000000000000000874B38C2E002DFF
12-13	
14	7100020002000000100AA82020100446F6F722053656E736F722020202020FFFB91
15	81000000010002000200AA840000000000000874B38C2E000000065BC31340004BE8
17-18	
18	7100020002000000100AA240000000000000000000000065BC3134000202E
19	81000000010002000200AA22010200536972656E20202020202020202020FFED06

The following happened at each step in Table 5.9 to Table 5.11:

1. Device 2, with short address 0x0100 and MAC address of 0x000000065BC31340, sends a Request connection to the coordinator. The ID of the endpoint is 0x01, it belongs to the security group, has a digital capability and is an input. The request key is 0x05.
2. Device 3, with short address 0x0200 and MAC address 0x0000000874B38C2E, sends a Request connection to the coordinator. The ID number of the endpoint is 0x02, it belongs to the security group, has a digital capability and is an 'AND' output. The request key is 0x05. The coordinator compares both requests and finds that their capabilities match.
3. The coordinator sends a Resolve MAC request to device 0x0100 confirming that its MAC address is still 0x000000065BC31340.
4. The coordinator sends a Resolve MAC request to device 0x0200 confirming that its MAC address is still 0x0000000874B38C2E.
5. Device 3 sends an Acknowledge MAC frame to the coordinator.
6. Device 1, the coordinator, sends an Acknowledge connection reply to Device 3.
7. Device 2 sends an Acknowledge MAC frame to the coordinator.
8. Device 1, the coordinator, sends an Acknowledge connection reply to Device 2. The coordinator function has now completed the establishing process.

9. The coordinator will now only act as a router for traffic between Device 2 and Device 3.  
Device 2 sends a Resolve MAC to Device 3 via the coordinator.
10. The coordinator forwards the request to Device 3.
11. Device 3 sends an Acknowledge MAC reply via the coordinator to Device 2.
12. The coordinator forwards the reply to Device 2.
13. Device 2 sends, via the coordinator, a Request endpoint status. This includes its description, door sensor and its on status.
14. The coordinator forwards the reply to Device 3.
15. Device 3 confirms that the MAC address of Device 2 is still valid.
16. The coordinator forwards the Request MAC address to Device 2.
17. Device 2 sends an Acknowledge MAC via the coordinator to Device 3.
18. The coordinator forwards the reply to Device 2.
19. Device 3 sends, via the coordinator, an Acknowledge endpoint status to Device 2. This includes its description, siren and its 'ON' status.
20. The coordinator forwards the reply to Device 2.

Figure 5.32 illustrates a snapshot of Device 2 after step 20. Steps 9 to 20 are repeated if the input status changes or any periodic updates are required. The Resolve MAC address may seem unnecessary at times, but this is also a way for the protocol to protect an endpoint against the spoofing of a short address.

### **5.6.3 Devices changes short addresses**

This simulation is the same setup as in 5.6.2, but Device 2 is now setup to only accept offers from a device with a short address 0x0Y00. This will simulate that Device 2 has moved out of the reception area of Device 1 and can only communicate with Device 3. The packet captures of Device 1 to 3 are shown from Table 5.12 to Table 5.14. Capturing starts after the devices have updated their endpoints.

Since steps have previously been discussed in detail, only groups of commands will be discussed from this point forward.

Table 5.12 Short addresses changes. Device 1: MAC 0000001641A75694

Step	Device 1: MAC 0x0000001641A75694, Short 0x0000
1	820001000000AA406F
2	820001000000AA406F
3	820001000000AA406F
4	820002000000AA193F
5	828000000200AA8A25
6-7	
8	8800FFFFFFFFFFFFFFFF0000000874B38C2E0000001641A75694AAFFFB7E
9	88400000000874B38C2E0000001641A756940000001641A75694AA0100B21B
10	88800000001641A756940000000874B38C2E0000001641A75694AA0100B8FF
11-19	
20	820001000000AA406F
21	828000000100AABA46
22-23	
24	820000000100AA677E
25	828001000000AA9D57
29	71000000020001000110AA84000000000000065BC313400000000874B38C2E003F30
30	61000200020000000110AA84000000000000065BC313400000000874B38C2E00D0EB
31-36	

Table 5.13 Short addresses changes. Device 2: MAC 000000065BC31340

Step	Device 2: MAC 0x000000065BC31340, Short 0x0100
1	8800FFFFFFFFFFFFFFFF000000065BC313400000001641A75694AAFFFF10BF
2	8840000000065BC31340000000874B38C2E0000001641A75694AA02108C9E
3	88800000000874B38C2E000000065BC313400000001641A75694AA0210B85C
	Device 2: MAC 0x000000065BC31340, Short 0x0210
4-5	
6	820002000210AA397D
7	828002100200AAE736
8-10	
11	820002000210AA397D
12	820002000210AA397D
13	81000200020002100210AA840000000000065BC31340000000874B38C2E00A3B4
14	8100FFFFFFFF02100210AA840000000000065BC31340000000874B38C2E003EAE
15	820002000210AA397D
16	
17	8800FFFFFFFFFFFFFFFF000000065BC31340FFFFFFFFFFFFFFFFFAFFFF8BD5
18	8840000000065BC31340000000874B38C2E0000001641A75694AA01108C9E
19	88800000000874B38C2E000000065BC313400000001641A75694AA0110B85C
	Device 2: MAC 0x000000065BC31340, Short 0x0110
20-21	
22	820001100100AA533D
23	828001000110AA8D76
24-25	
26	820001100100AA533D



The following functions occur at each step in Table 5.12 to Table 5.14

- 1 to 3. Device 2 moves away from Device 1, the coordinator's, range and requests a short address. Device 2 accepts short address 0x0210 from Device 3. The coordinator sends a Request Keepalive, but does not receive any replies. The coordinator clears its neighbour address 0x0100.
- 4 to 5. Keepalive requests and acknowledgements are sent between the coordinator and Device 3. Only one cycle is shown, but step 4 and 5 are repeated until communication is lost between Device 1 and Device 3.
- 6 to 7. Device 2 and Device 3 send Keepalive requests and acknowledgements.
- 8 to 10. Device 3 requests a better short address and accepts the short address offer 0x0100 made by the coordinator. Device 2 will ignore a short address request from its parent.
- 11 to 12. Device 2 sends a Request Keepalive to the old short address of Device 3 without any reply.
13. Device 2 needs to send a Request endpoint status and sends a unicast Resolve MAC address to short address 0x0200 and does not receive any reply.
14. Device 2 sends a broadcast Resolve MAC address.
15. Device 2 sends a Request Keepalive to its parent device 0x0200 and does not receive any reply because the short address of Device 3 is now 0x0100. This was the third attempt and this triggers Device 2 to request a short address.
16. Device 3 sends an Acknowledge MAC address reply to Device 2 with short address 0x0210. Device 2 does not receive the packet because it is in the process of requesting a new short address.
- 17 to 19. Device 2 requests a new short address and receives only an offer from Device 3 and accepts it.
- 20 to 21. Keepalive sent between Device 1 and Device 3.
- 22 to 23. Keepalive sent between Device 2 and Device 3.

- 24 to 25. Keepalive sent between Device 1 and Device 3.
- 26 to 27. Keepalive sent between Device 2 and Device 3.
- 28 to 30. Device 2 needs to confirm that the device with the MAC address of 0x0000000874B38C2E, still has a short address of 0x0200. The packet is forwarded to Device 3, but it will not investigate the Payload, even if the MAC address does belong to it, and forwards the frame to the coordinator. The coordinator forwards the frame to 0x0200. There is no device with that short address and the Resolve MAC address timer of Device 2 expires.
- 31 to 32. Device 2 broadcasts to all devices a Resolve MAC address request and Device 3 replies with an Acknowledge MAC address.
- 33. Device 2 sends a Request endpoint status to Device 3 that includes its endpoint description and status.
- 34 to 36. Device 3 sends an Acknowledge endpoint status reply to Device 2.

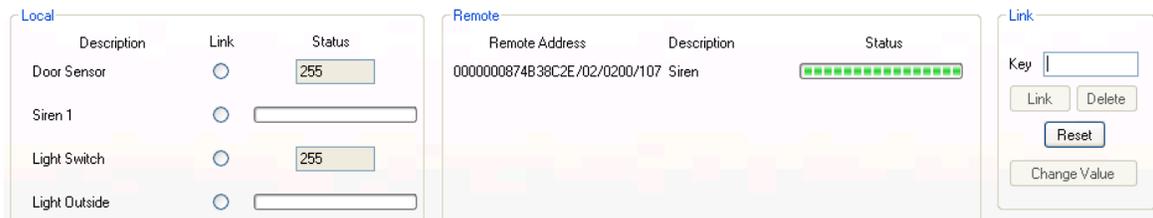


Figure 5.33 Snapshot of Device 2 before short address changes

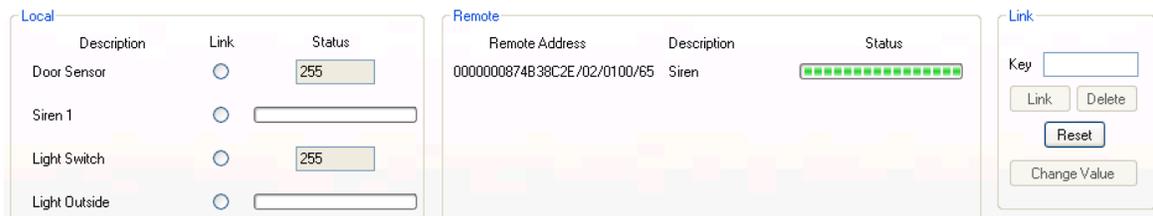


Figure 5.34 Snapshot of Device 2 after the short address changes

Figure 5.33 illustrates Device 2 with a connection established. This diagram illustrates that the remote endpoint ID of endpoint 1 is 0x02 and the short address of the remote hosting device is 0x0200. After the devices have changed their short addresses as decrypted above, Figure 5.34 illustrates that Device 2 has discovered the new short address of Device 3.

### 5.6.4 One output endpoint linked to two input endpoints

In this simulation, three devices are set up. One device has a siren endpoint with ‘AND’ output capabilities. An input endpoint is linked with each of the other devices. The devices have already discovered their short addresses and all Keepalive commands are filtered out. This simulation also illustrates how Device 1 initiates an auto endpoint update and how Device 3 manually changes its endpoint status.

Table 5.15 Three endpoints. Device 1: MAC 0000001641A75694

Step	Device 1: MAC 0x0000001641A75694, Short 0x0000, Door Sensor Endpoint 1
1	8100000000001000100AA8802011300000000000000000000065BC31340051DE5
2	Coordinator send internal Request connection for local endpoint 1
3	81000100010000000000AA840000000000001641A75694000000065BC3134000C4CA
4	8100000000001000100AA2400000000000000000000000000065BC31340001C51
5	81000100010000000000AA280101110000000000000000000001641A75694053D29
6	Coordinator stores Device 2 Endpoint information
7	81000100010000000000AA82020100446F6F722053656E736F722020202020FF15C0
8	8100000000001000100AA84000000000000065BC31340000001641A7569400C181
9	81000100010000000000AA240000000000000000000000000001641A75694008EF5
10	8100000000001000100AA22010200536972656E2031202020202020202020FF16D7
11	
12	7100000000001000110AA88070111000000000000000000000874B38C2E07ABC2
13	8100000000001000100AA8802011300000000000000000000065BC31340071DE5
14	81000100010000000000AA840000000000001641A75694000000874B38C2E00ADAC
15	
16	81000100010000000000AA840000000000001641A75694000000065BC3134000C4CA
17	8100000000001000100AA2400000000000000000000000000065BC31340001C51
18	
19	7100000000001000110AA24000000000000000000000000000874B38C2E00EA89
20	81000100010000000000AA2807011101100000000000000000874B38C2E0767EE
21	81000100011000000000AA2802011301000000000000000000065BC31340078880
22-26	
27	8100000000001000100AA84000000000000065BC31340000001641A7569400C181
28-29	
30	81000100010000000000AA240000000000000000000000000001641A75694008EF5
31	8100000000001000100AA22010200536972656E20312020202020202020200016D7
	Device 1 auto update endpoint status

Table 5.15(Continuous) Three endpoints. Device 1: MAC 0000001641A75694

32	81000100010000000000AA840000000000001641A7569400000065BC3134000C4CA
33	81000000000001000100AA2400000000000000000000000000065BC31340001C51
34	81000100010000000000AA82020100446F6F722053656E736F722020202020FF15C0
35	81000000000001000100AA84000000000000065BC31340000001641A7569400C181
36	81000100010000000000AA2400000000000000000000000000001641A75694008EF5
37	81000000000001000100AA22010200536972656E203120202020202020200016D7
Device endpoint status changes from OFF to ON	
38-41	
42	81000000000001000100AA84000000000000065BC31340000001641A7569400C181
43	
45	81000100010000000000AA2400000000000000000000000000001641A75694008EF5
46	81000000000001000100AA22010200536972656E20312020202020202020FF16D7

Table 5.16 Three endpoints. Device 2: MAC 000000065BC31340

Step	Device 2: MAC 0x00000065BC31340, Short 0x0100, Siren Endpoint 2
1	81000000000001000100AA8802011300000000000000000000065BC31340051DE5
2	
3	81000100010000000000AA840000000000001641A7569400000065BC3134000C4CA
4	81000000000001000100AA2400000000000000000000000000065BC31340001C51
5	81000100010000000000AA2801011100000000000000000000001641A75694053D29
6	
7	81000100010000000000AA82020100446F6F722053656E736F722020202020FF15C0
8	81000000000001000100AA84000000000000065BC31340000001641A7569400C181
9	81000100010000000000AA2400000000000000000000000000001641A75694008EF5
10	81000000000001000100AA22010200536972656E203120202020202020200016D7
11	81000100000001100110AA88070111000000000000000000000874B38C2E07B597
12	71000000000001000110AA88070111000000000000000000000874B38C2E07ABC2
13	81000000000001000100AA8802011300000000000000000000065BC31340071DE5
14	81000100011000000000AA840000000000001641A75694000000874B38C2E00ADAC
15	71000110011001000000AA840000000000001641A75694000000874B38C2E00D659
16	81000100010000000000AA840000000000001641A7569400000065BC3134000C4CA
17	81000000000001000100AA2400000000000000000000000000065BC31340001C51
18	81000100000001100110AA24000000000000000000000000000874B38C2E00F4DC
19	71000000000001000110AA24000000000000000000000000000874B38C2E00EA89
20	81000100010000000000AA28070111011000000000000000000874B38C2E0767EE
21	81000100011000000000AA2802011301000000000000000000065BC31340078880
22	71000110011001000000AA2802011301000000000000000000065BC3134007F375
23	81000100010001100110AA840000000000000874B38C2E000000065BC31340003EAA
24	81000110011001000100AA2400000000000000000000000000065BC3134000C921
25	81000100010001100110AA8202070057696E646F7753656E736F7220526F6F00C9A7
26	81000110011001000100AA84000000000000065BC31340000000874B38C2E0026D9
27	81000000000001000100AA84000000000000065BC31340000001641A7569400C181
28	81000100010001100110AA24000000000000000000000000000874B38C2E0058BD
29	81000110011001000100AA22070200536972656E2031202020202020202000C343
30	81000100010000000000AA2400000000000000000000000000001641A75694008EF5
31	81000000000001000100AA22010200536972656E203120202020202020200016D7
Device 1 auto update endpoint status	
32	81000100010000000000AA840000000000001641A7569400000065BC3134000C4CA
33	81000000000001000100AA2400000000000000000000000000065BC31340001C51
34	81000100010000000000AA82020100446F6F722053656E736F722020202020FF15C0
35	81000000000001000100AA84000000000000065BC31340000001641A7569400C181
36	81000100010000000000AA2400000000000000000000000000001641A75694008EF5

Table 5.16(Continuous) Three endpoints. Device 2: MAC 000000065BC31340

37	8100000000001000100AA22010200536972656E203120202020202020200016D7
	Device endpoint status changes from OFF to ON
38	81000100010001100110AA84000000000000874B38C2E000000065BC31340003EAA
39	81000110011001000100AA2400000000000000000000000000065BC3134000C921
40	81000100010001100110AA8202070057696E646F7753656E736F7220526F6FFFC9A7
41	81000110011001000100AA84000000000000065BC31340000000874B38C2E0026D9
42	8100000000001000100AA84000000000000065BC31340000001641A7569400C181
43	81000100010001100110AA24000000000000000000000000000874B38C2E0058BD
44	81000110011001000100AA22070200536972656E20312020202020202020FFC343
45	8100010001000000000AA24000000000000000000000000001641A75694008EF5
46	8100000000001000100AA22010200536972656E20312020202020202020FF16D7

Table 5.17 Three endpoints. Device 3: MAC 0000000874B38C2E

Step	Device 3: MAC 0x0000000874B38C2E, Short 0x0110, Window Sensor Endpoint 5
1-10	
11	81000100000001100110AA8807011100000000000000000000000874B38C2E07B597
12	
15	71000110011001000000AA84000000000001641A75694000000874B38C2E00D659
16-17	
18	8100010000001100110AA24000000000000000000000000000874B38C2E00F4DC
19-21	
22	71000110011001000000AA280201130100000000000000000000065BC3134007F375
23	81000100010001100110AA84000000000000874B38C2E000000065BC31340003EAA
24	81000110011001000100AA2400000000000000000000000000065BC3134000C921
25	81000100010001100110AA8202070057696E646F7753656E736F7220526F6F00C9A7
26	81000110011001000100AA84000000000000065BC31340000000874B38C2E0026D9
27	
28	81000100010001100110AA24000000000000000000000000000874B38C2E0058BD
29	81000110011001000100AA22070200536972656E2031202020202020202000C343
30-37	
	Device endpoint status changes from OFF to ON
38	81000100010001100110AA84000000000000874B38C2E000000065BC31340003EAA
39	81000110011001000100AA2400000000000000000000000000065BC3134000C921
40	81000100010001100110AA8202070057696E646F7753656E736F7220526F6FFFC9A7
41	81000110011001000100AA84000000000000065BC31340000000874B38C2E0026D9
43	81000100010001100110AA24000000000000000000000000000874B38C2E0058BD
44	81000110011001000100AA22070200536972656E20312020202020202020FFC343

The following functions occur at each step in Table 5.15 to Table 5.17:

- 1 to 2. Device 2 transmits its Request connection to the coordinator. The command specifies the following: Local ID is 2, it is part of the security group, it has digital capability and the IO type is 'AND' output. The coordinator internally registers its request and specifies the following: Local ID is 1, it is part of the security group, it has digital capability and has an input IO type. Both request keys are 5 and the coordinator links them together.
- 3 to 6. The coordinator confirms the MAC address and short address of Device 2 and transmits an Acknowledge connection to Device 2 containing the endpoint information of Device 1. The coordinator stores the endpoint information of Device 2 in its Remote endpoint table.
7. Device 1 sends a Request endpoint status to Device 2 containing its endpoints description and the 'ON' status.
- 8 to 10. Device 2 confirms the MAC address and short address of Device 1 and sends an Acknowledge endpoint status reply containing its endpoint description and confirms that the output status is 'ON'. Figure 5.35 illustrates a snapshot of Device 2 after step 10 is completed.



Figure 5.35 Snapshot of Device 2 after first connection is established

- 11 to 12. Device 3 transmits a Request connection via Device 2 to the coordinator. The command specifies the following: Local ID is 5, it is part of the security group, it has digital capability, it has an input IO type and has a key of 7.
13. Device 2 transmits a Request connection to the coordinator. The command specifies the following: Local ID is 2, it is part of the security group, it has digital capability, it has an 'AND' output IO type and it has a key of 7. The coordinator recognises that both requests are partners.
- 14 to 19. The coordinator confirms that both devices and short addresses belong to the requester's MAC address and receives acknowledgements from both devices.
- 20 to 22. The coordinator sends an Acknowledge connection to both devices containing the endpoint information of the remote device.
- 23 to 25. Device 3 confirms the MAC address of Device 2 and sends a Request endpoint status that contains its endpoint description and its 'OFF' status. The IO type of Device 2 is an 'AND' output and it calculates that its output must be 'OFF'.
- 26 to 27. Device 2 sends a Request MAC address to the hosting devices of both remote endpoints. A request is sent to Device 1 because the output status of Device 2 has changed.
- 28 to 29. Device 3 acknowledges the Resolve MAC address request of Device 2. Device 2 sends an Acknowledgement endpoint status reply containing its description and status. Device 3 updates the remote indication of endpoint 2. Figure 5.36 illustrates a snapshot of the simulation of Device 2 after the second connection have been established.

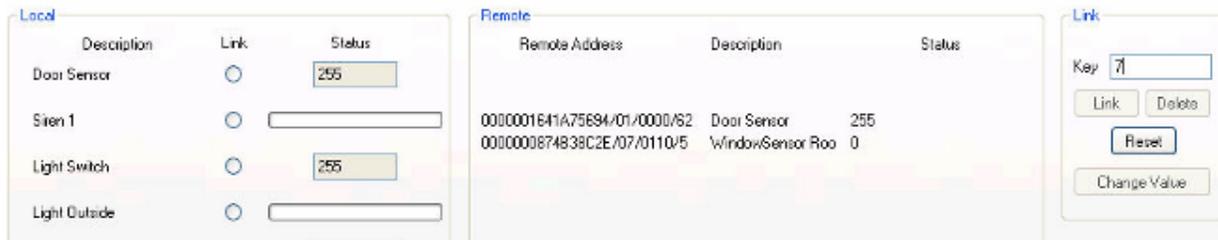


Figure 5.36 Snapshot of Device 2 after second connection is established

30 to 31. Device 1 acknowledges the Resolve MAC address request of Device 2. Device 2 sends an Acknowledgement endpoint status reply containing its description and status. Figure 5.37 is a snapshot of Device 1 illustrating that the remote indication of endpoint 1 shows 'OFF'.

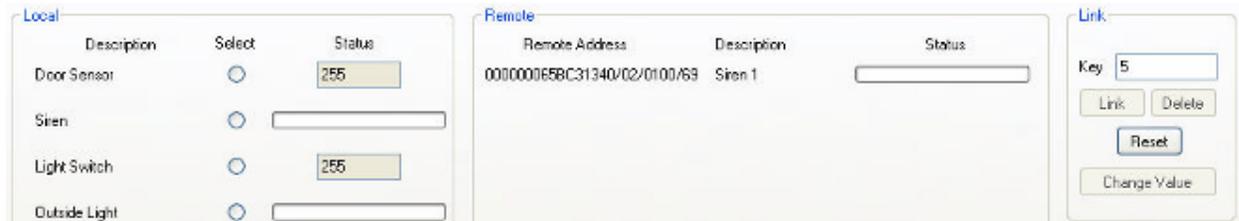


Figure 5.37 Snapshot of Device 1 after second connection is established

32 to 34. The Remote endpoint 1 timer of Device 1 expires and it initiates an auto Request endpoint status. The Mac address and short address is confirmed before the request is sent.

35 to 37. Device 1 receives the Acknowledge endpoint status of Device 2. The status of all the endpoints remains the same.

38 to 40. The endpoint status of Device 3 changes from 'OFF' to 'ON'. It informs Device 2 with a Request endpoint status command.

41 to 42. Device 2 calculates that its output must also change from ‘OFF’ to ‘ON’ and must not only send an acknowledgement to Device 3, but also must inform Device 1 of the status change and send Resolve MAC addresses to both hosting devices of the remote endpoints. Figure 5.38 illustrates a snapshot of Device 2 after the calculation has been done.



Figure 5.38 Snapshot of Device 2 after Device 1's endpoint status changes

43 to 44. Device 3 receives an acknowledgement and updates its remote indication.

45 to 46. Device 1 receives an acknowledgement and updates its remote indication.

## 5.6.5 Analogue endpoints

In this simulation two devices are used. One endpoint is a dimmer switch and the other a light bulb. The connection was already established and the dimmer value was ‘OFF’. The simulation starts when the dimmer value changes to 100.





## 5.7 Conclusion

The functions mentioned in this chapter operate above the Discovery suite, including the short address and Keepalive functions.

The Resolve MAC address suite is responsible for the link between the Discovery suite and the set up and maintenance of connections and endpoint statuses. It also verifies that the remote device is not a device spoofing a short address.

A connection can be established between endpoints with the same characteristics. Endpoints with an 'AND' or 'OR' IO type can have two input endpoints. These endpoints can be on the same device or multiple devices and the coordinator can host the endpoints.

The Source and Destination short address fields make it possible to route endpoint traffic to non-directly connected devices without a routing table. The Resolve MAC address suite will send a broadcast if the short address of a device has changed. To eliminate a broadcast storm, a device cannot receive a broadcast that they have sent in a Wireless Mesh topology.

When a device receives a Resolve MAC address it will suspend the function to discover a better short address. This allows the command following to communicate to the short address of the remote devices. The protocol was designed to allow devices to move around. If a solution does not require mobility, the Resolve MAC address function does not need to be sent before each command, but only if a command does not receive a reply.

The only manual interaction required is when a new connection is established between endpoints. This is to select a key and the developer must develop an interface for the user to enter the key. If a device has more than one endpoint the developer must also develop an interface to select which endpoint it needs to connect or disconnect.

# Chapter 6

## 6 Constructing of the device controller

This chapter will discuss the physical development of the Device controller. The decision was made from the start to develop the device without any wireless connection. A Universal Synchronous Asynchronous Receiver Transmitter (USART) interface connected to a Personal Computer (PC) was utilised. This simplifies the development of the microprocessor software and resolves software problems eliminating any radio-associated problems. The PC will be connected to another PC via Ethernet to simulate many devices communicating to the device controller, as discussed in the Simulate wireless section in section 4.1. Finally, the serial interface is replaced with a wireless transceiver.

### 6.1 Block diagram

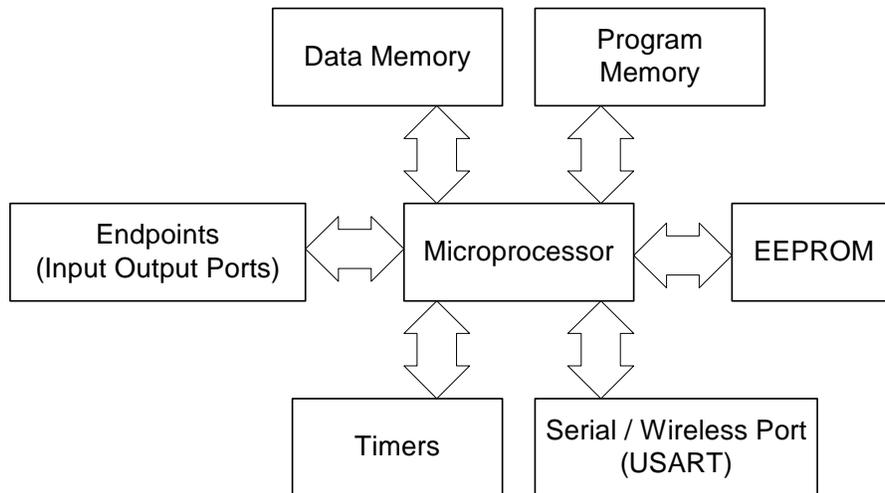


Figure 6.1 Controller basic block diagram requisites

Figure 6.1 illustrates the basic components needed by the device. The device will need a microprocessor, Read Only Memory (ROM) for the program and Random Access Memory (RAM)

for the variables. The device will also need IO ports where the endpoints would be defined and Electrically Erasable Programmable Read Only Memory (EEPROM) to store all the established endpoint connections. The USART will be used for communication to other devices or simulated devices. The medium must be a serial connection or a wireless transceiver. The Timer will be used to perform scheduled tasks.

## 6.2 Choosing the correct components

Choosing the correct microprocessor for a project can become a time-consuming process, since there are a variety of vendors to choose from. In addition, each vendor has a variety of products to choose from.

The microchip-microprocessor product range compliments products in the same range. If a microprocessor is used and later on the design requires more RAM, then a different microprocessor with the same feature set, pin layout and the required RAM, can be used without changing the micro code. For this reason it was decided to use the PIC18LF462 processor. It has 65536 FLASH, 3968 RAM, 1024 Data EEPROM, 4 Timers, 5 I/O ports and an enhanced USART. The PIC18LF462 processor supports in-circuit serial programming and in-circuit debugging [12].

Figure 6.2 illustrates the Microchip MPLAB<sup>®</sup> ICD programmer. The MPLAB<sup>®</sup> software can be downloaded free from the Internet. This programmer allows the developer to do in-circuit debugging of the project.



Figure 6.2 Microchip MPLAB<sup>®</sup> ICD2 programmer

Another tool that was used to develop the hardware was the PIC18 Simulator IDE. It allows the micro code to be developed in Basic and compiles a Hex and Assembler files needed by the ICD2 programmer to program the microprocessor. Another big advantage is that the PIC18 Simulator can simulate eight LED's, Matrix Keypad, LCD, Graphical LCD and many more. The hardware USART interface was used in this project. Figure 6.3 illustrates a screenshot of the software.

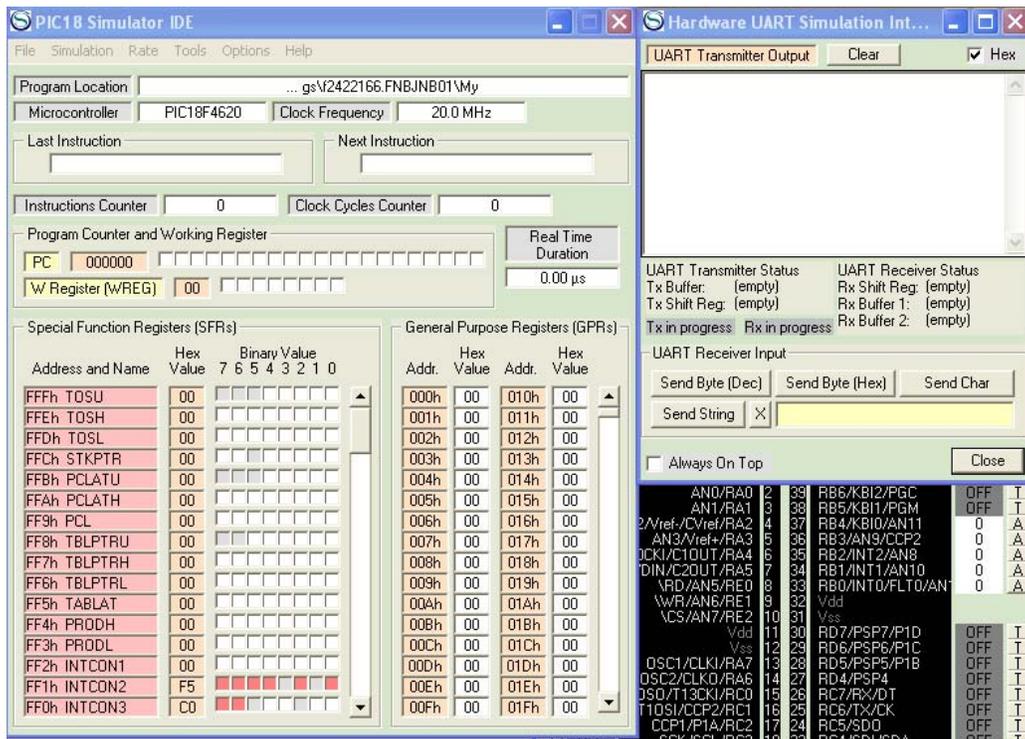


Figure 6.3 PIC18 Simulator IDE

The second important component used in the project is the wireless transceiver. The RFD21733 transceiver was chosen, because it was one of the most economical transceivers available and it had a 9600-baud half-duplex USART interface and an indoor range of 33 meters [16]. An important electrical specification was that the maximum voltage of the component is 3.6 volts and that a 3.3V to 5V logical level shifter must be incorporated into the design [16]. Although the wireless transceiver supports multi-point-to-multi-point configuration, it was decided not to implement this feature,

because the functionality of the protocol needs to be tested in a non-intelligent wireless implementation.

### 6.3 Developing the PIC18LF462 microcontroller software

Figure 6.4 illustrates the laboratory setup. A serial connection from one PC via Ethernet to another PC is used. This setup is used to develop the first phase of the PIC18LF462 micro code. The PC simulation software for the first PC will act as a Serial to Ethernet translator, and vice versa. This will happen when the Serial and Ethernet selections are in use as discussed in the Global setup section in section 4.2.2.

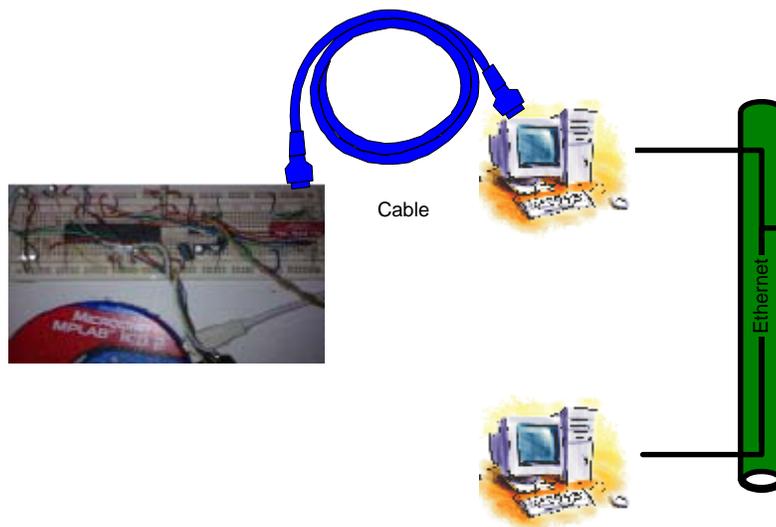


Figure 6.4 Micro hardware device setup first phase

#### 6.3.1 Interrupt handling

The software was written identical to the flow diagrams shown in Chapter 4 and 5. The first problem experienced was that the random CRC calculation of the received framed would be incorrect. The controller will now incorrectly discard these packets. After some investigation it was discovered that

the incorrect calculation occurred after a timer interruption. This problem was resolved by saving the following register content as the Microchip datasheets suggests during interrupt service routines:

- WREG,
- STATUS,
- BSR [17].

This is illustrated in the code segment below.

```
ASM:  movff  W,      w_temp
ASM:  movff  STATUS, status_temp
ASM:  movff  BSR,    bsr_temp

<Interrupt routine>

ASM:  movff  w_temp,  w
ASM:  movff  status_temp, STATUS
ASM:  movff  bsr_temp, BSR
```

### 6.3.2 Configuring the USART

The Device discovery and maintenance layer was written in Basic and successfully tested. Next a connection between a simulated endpoint on the PC and the device must be established. When a second PC was added via an Ethernet connection, and a connection was established between the device and the second PC, the device would randomly stop to reply on requested packets.

Investigation verified that the Buffer overrun bit (RCSTA<OERR>), was set and it caused the USART to stop responding. To solve the problem Microchip recommends that the USART receive buffer (PIR1<RCIF>) interrupt routine must be initialised to test the overrun error bit (RCSTA<OERR>). If there is an error condition, it can be reset by cycling the Continuous Receive

Enable bit (RCSTA<CREN>). After removing the error condition, the USART receive (RCREG) register must be read until the PIR1<RCIF> bit is cleared [18].

The code segment is illustrated below.

```
    If PIR1.RCIF = True Then           // Receive USART interrupt
        PIE1.RCIE = False             // Reset USART interrupt flag
        usartrd:
        If RCSTA.OERR = True Then     //Is there a Buffer overrun
            RCSTA.CREN = False       // Cycle the CREN bit
            RCSTA.CREN = True
        Endif
        Gosub rxbuffer                // Call the function that reads the USART
        If PIR1.RCIF = True Then Goto usartrd
    Endif
```

This resolved the problem when the device completely stopped responding to requests. However the device would still randomly be unresponsive to requests. The above solution did not resolve the buffer overrun error, it was only managing the error and the request would still be corrupted. Microchip recommended that no single interrupt routine must last longer than 1/10 of the USART data byte reception period [18].

The device USART is configured for 9600-baud rate. No interrupt routine must be longer than:

$$\begin{aligned} \text{Max time} &= 1/10 \times 9600 \text{ bits}/8 && (4) \\ &= 833 \text{ us} \end{aligned}$$

The PIC18LF462's instruction cycle = 4 clock cycles [17] and is clocked by a 20 MHz crystal and thus the

$$\begin{aligned} \text{Instruction time} &= 1/20 \text{ MHz} \times 4 && (5) \\ &= 2 \text{ us} \end{aligned}$$

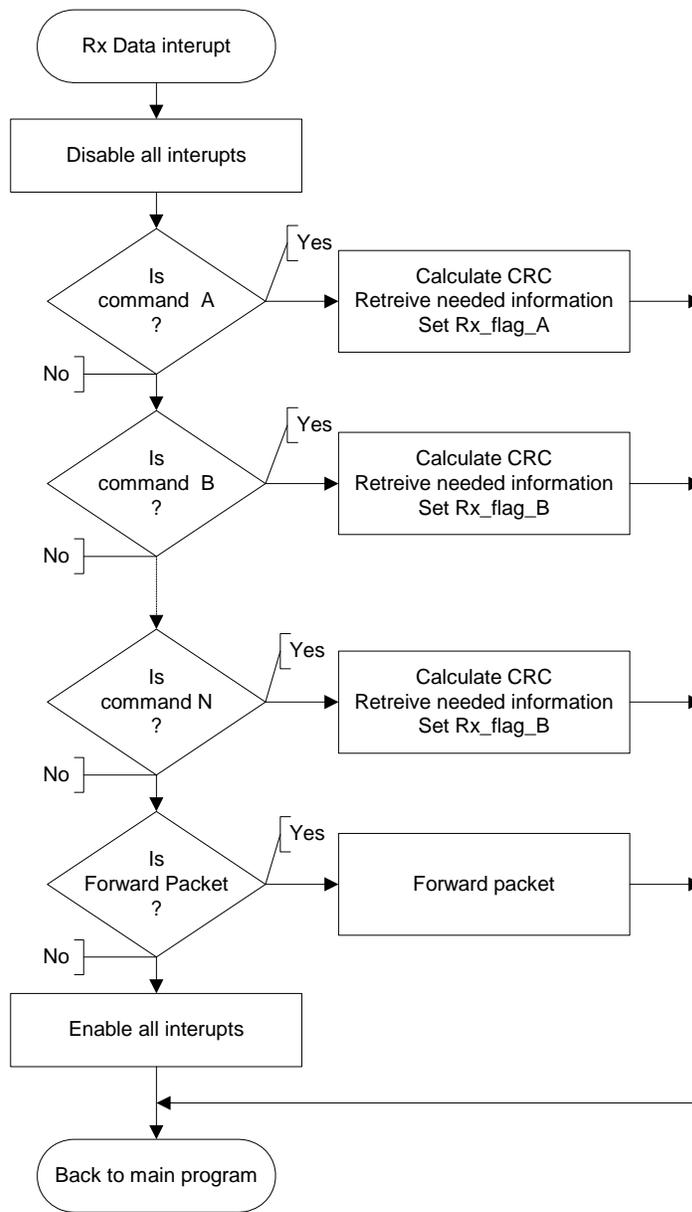


Figure 6.5 Receive USART interrupt routine

No RX interrupt routine must **exceed 416 instructions**. To accomplish this, the complete Receive data interrupt routine was redesigned. When data is received, the microprocessor will investigate the first byte and if it is valid data, store it. Valid values are defined as 0xY8, 0xY2 and 0xY1.

The software will raise an alert when the complete command was received. The length will depend on the command suite. Previously the devices would start executing the request, but as illustrated in Figure 6.5 the device will now first calculate the CRC value and if correct, store the data in registers to be analysed at a later stage. In addition an associated flag is raised. If the packet is not destined for this device, it will be forwarded during the interrupt routine.

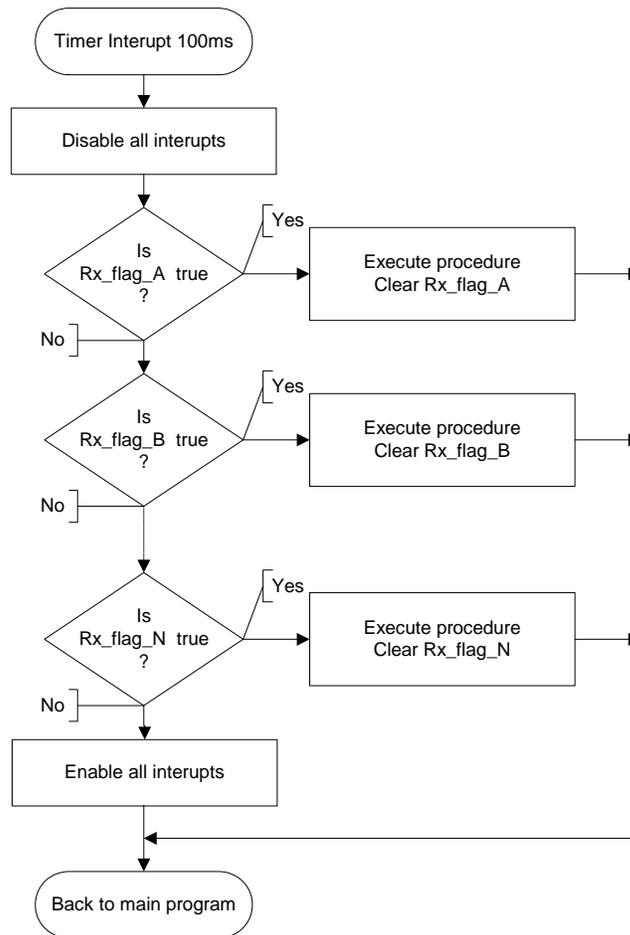


Figure 6.6 Timer interrupt routine for data received

During the first timer interrupt, after a data packet has been received, all the different command flags are inspected. If any of the flags are set, they will be processed as illustrated in Figure 6.6. It is

important to note that although the processor is busy serving an interrupt, all other interrupts are disabled.

After the above fixes have been applied, buffer overrun errors were limited, but still occurring. From a user's perspective everything was working appropriately, but it was a concern that retransmissions occurred, with only three devices.

### **6.3.3 Redesign the Request connection process**

In section 5.7 it was mentioned that the role of the Resolve MAC addresses could be changed if the device is not mobile. It was decided to redesign the Add connection and Update endpoint status process as discussed in section 3.2.3.2 and section 3.2.3.3.

It was decided to minimise the data transmitted when a new connection is established between two endpoints. As illustrated previously in Figure 3.19, it took eight commands to build a connection. Another problem with the process was that if one of the acknowledgements from the coordinator gets corrupted, the connection is established from only one device. The modification must allow the device to send an acknowledgment back to the coordinator and allow the coordinator to retransmit the acknowledgement. The Connection Control field, illustrated in Figure 5.1, was changed to identify between Connection commands send to the coordinator and Connection commands to the end device.

Figure 6.7 and Figure 6.8 illustrates the difference between the old and new Connection Control field. Field 0, previously reserved, will be used to identify packets sent to the coordinator and field 3 to the endpoint. Two different commands were needed because the information inside an acknowledgement transmitted differs in relation to the coordinator and to the endpoints.

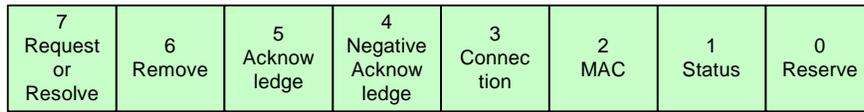


Figure 6.7 Old connection control field

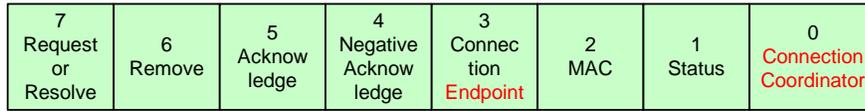


Figure 6.8 New connection field

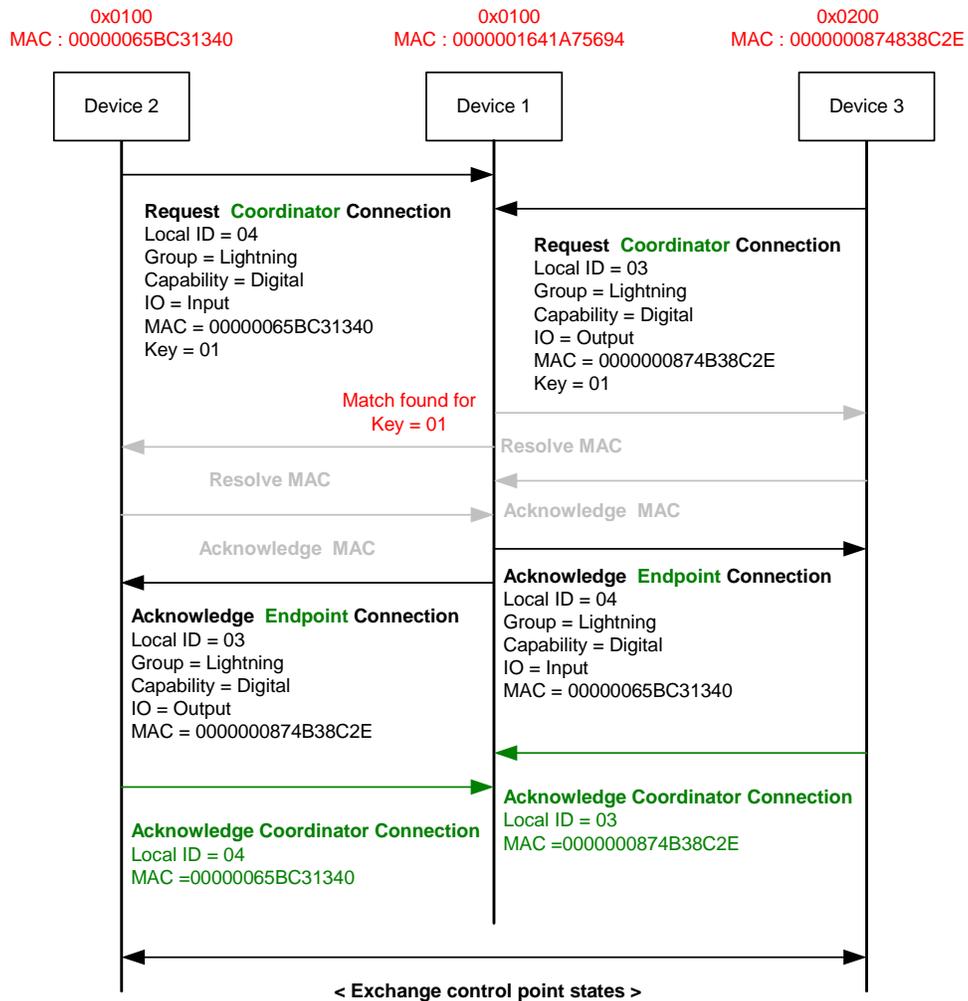


Figure 6.9 Add connection process modification successful

Both devices still send a Request Connection command, now called Request coordinator Connection, to the coordinator. When the coordinator identifies a matching request, it sends an Acknowledge Endpoint Connection to the device. Previously, a resolve MAC command would be send first. The coordinator will delete the Acknowledge Endpoint Connection after the reply is received as illustrated in Figure 6.9.

Figure 6.10 illustrates the new process when a reply for an Acknowledge Endpoint Connection is not received. It is important to note that the coordinator will attempt to contact Device 3 three times.

Previously a successful Add connection took eight commands, but the same process currently takes six commands. This is translated in a reduction in transmission of forty-eight bytes or a saving of 25%. After a device's short address has changed the rediscover of the new short address of the device, previously took nine commands, but with the new process it took ten commands. The increase of 10% in traffic has resulted in better reliability.

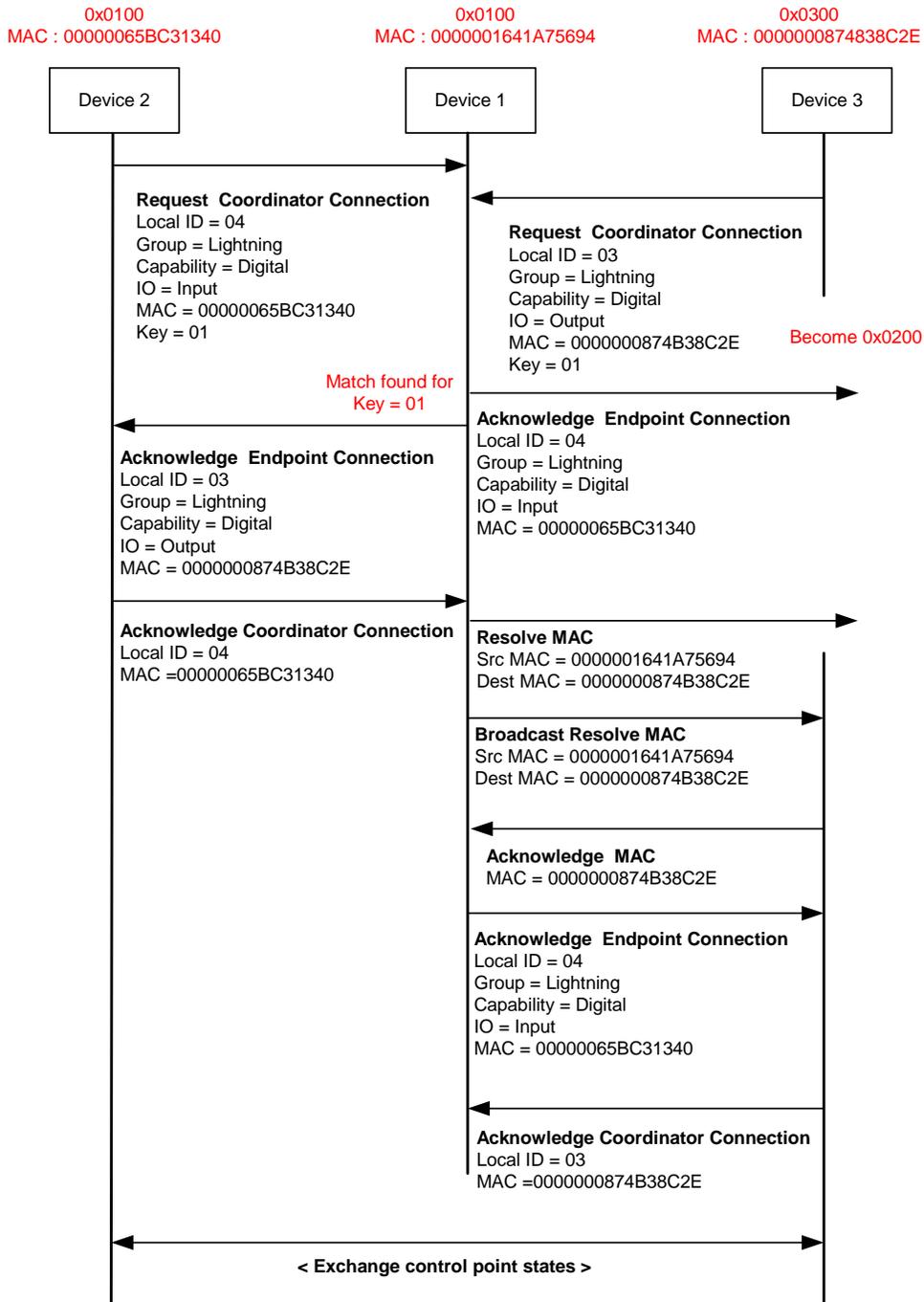


Figure 6.10 Add connection process modification unsuccessful

- Request coordinator connection modification



Figure 6.11 Request connection modified frame sample

Figure 6.11 illustrates the modified Request connection frame that was discussed in Figure 5.12. The Connection Control frame has changed from a Request connection, value 0x88, to a Request coordinator connection with a value of 0x81.

- Acknowledge endpoint connection

There is no difference between the Acknowledge connection requests discussed in Figure 5.14 and the newly named Acknowledge endpoint connection request.

- Acknowledge coordinator connection

The Acknowledge endpoint transfers all the capabilities of the remote endpoints to the local endpoint. The function of the Acknowledge coordinator connection is to acknowledge the Acknowledge endpoint connection.

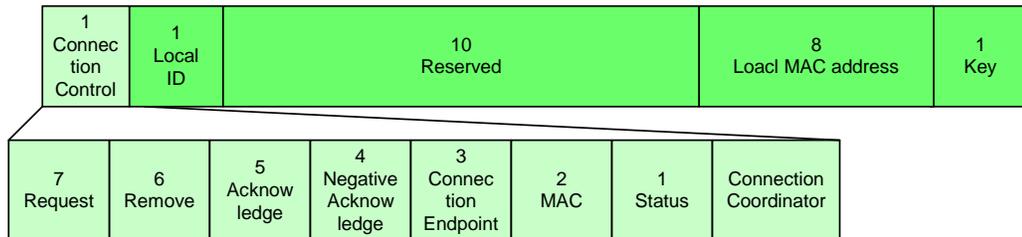


Figure 6.12 Acknowledge coordinator connection frame

Figure 6.12 illustrates the new Acknowledge coordinator connection frame format.

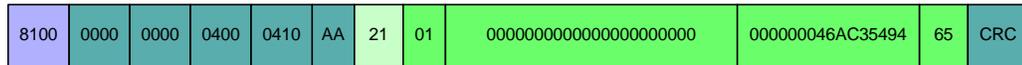


Figure 6.13 Acknowledge coordinator connection sample frame

Figure 6.13 illustrates device 0x0410 sending an Acknowledge coordinator connection reply to the coordinator via device 0x0400. The frame specifies the endpoint ID, MAC address and key of the local device that were used to establish the connection. The coordinator uses this information to link the reply to the related request.

### 6.3.4 Redesign the Manage connection process

With this process the same commands, Request status and Acknowledge status, were used. The key modification with this process was that the device would first transmit a Request endpoint status command. If an Acknowledge endpoint status is not received, a Resolve MAC request is sent. The process is the same after the Resolve MAC request is sent as discussed in section 3.2.3.3. As discussed in Figure 5.29a, the device will request an update when the endpoint status timer exceeds 120 seconds. The process was modified to send a Request endpoint status every 120 seconds. If an Acknowledge endpoint status is not received when the timer reaches 130 seconds, the Resolve MAC process will follow. A Request endpoint status command is transmitted after a successful Resolve MAC request. If the above-mentioned was unsuccessful and the timer exceeds 140 seconds, then the timer is changed to 130 seconds and the process repeats itself until a status update is received.

A problem would occur if the short address of Device A changes and Device B receives the old address and the same endpoint is active. If a status update were requested, the incorrect device would reply. To resolve the problem, the Request endpoint status command must be able to identify the MAC address of the remote device.



Figure 6.14 Request endpoint status modified frame

All the fields in the Request endpoint status command are already in use, except for the fourth field. The odds were less that the least significant bit (lsb) of the MAC addresses of the two devices would differ than the most significant bit (msb). For this reason it was decided to only add the lsb of the remote device MAC address as a verification method that the remote device MAC address did not change, as illustrated in Figure 6.14.

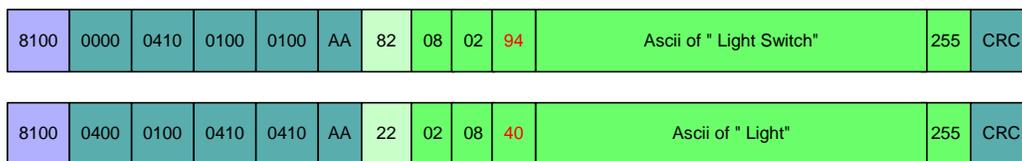


Figure 6.15 Request endpoint status modified sample frame

Figure 6.15 illustrates that the MAC address of Device 0x0410 must end with 0x94 and the xMAC address of Device 0x0100 must end with 0x40. This solution is not clear-cut, but the chances are slim for an incorrect device with the same endpoint ID to have a remote endpoint to another device with the same lsb MAC as the original request.

Example:

Device A, short address 0x0100, MAC address 0x1111222233334494, endpoint 0x01 has an endpoint connection to Device B, short address 0x0200, MAC address 0x1111222233334490, endpoint 0x02.

Device C, short address 0x0300, MAC address 0x5555666677778894, endpoint 0x01 has an endpoint connection to Device B, short address 0x0400, MAC address 0x5555666677778890, endpoint 0x02.

Device A loses connectivity and Device C request a new short address and receives address 0x0100. In this example the solution would not work, but the change for this to occur is negligible small.

### 6.3.5 Contact de-bouncing

The device also needed to eliminate the effect of contact bouncing of any of the input switches. An RC filter with a RC value of 0.1 seconds or an SR Latch can be used to do contact de-bouncing [19]. It was decided to implement the de-bouncing in the software as alternative.

```
Symbol bt_addlink = PORTD.6

Main:
    If bt_add = 0xff Then                //Switch on flag set
        {Do on procedure}
    else if bt_add = 0x00 then          //Switch off flag set
        {Do off procedure}
    endif
    goto Main                            //Endless loop

Timer_Interupt:
    If PIR1.TMR1IF = True Then          // 10msec timer interupt

        'Timer1 10msec
        bt_add = ShiftLeft(bt_add, 1)   //Shift Switch byte value
        bt_add.0 = bt_addlink           //Add new shift bit value
    Endif
Return
```

The above code segment illustrates software de-bouncing. PORTD.6 is read as a bit value every 10 milliseconds. The value is stored as a byte value. The byte value is shifted left and the new value is added to bit 0. This means it will take bt\_add variable 80 milliseconds to reach the value 11111111b, after the contact bouncing becomes stable. When the switch contact is still vibrating the value of bt\_add will be between 0x00 and 0xff and the software will ignore this state. The 80 milliseconds is longer than the required 40 milliseconds for contact debouching.

### 6.3.6 Device circuit diagram

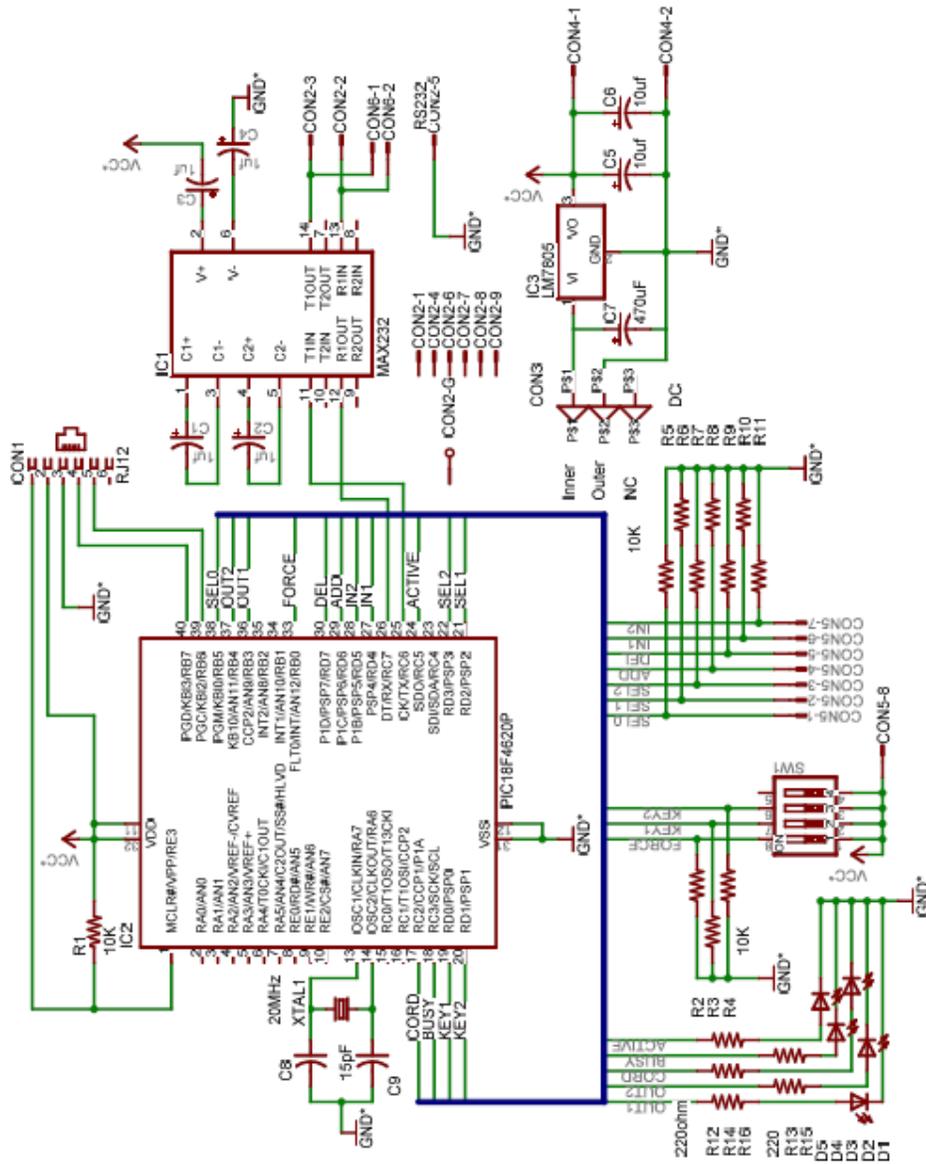


Figure 6.16 Device circuit diagram

Figure 6.16 illustrates the circuit diagram for the controller board. The controller board has a MAX-232 Integrated Circuit (IC) connected to the Micro controller’s USART. It will be used to communicate with the simulator and later via the wireless modems. The minimum wires needed between two MAX-232 IC’s is Transmit data, Receive Data and Ground. The controller is designed as

a Data Terminal Equipment (DTE) and the wireless transceiver as a Data Communication Equipment (DCE). The device can be connected directly to the simulator PC through a 9-pin crossover cable or with a straight cable to the wireless transceiver. The connecting diagram is shown in Table 6.1 and Table 6.2.

*Table 6.1 3 Wire Straight Cable*

DTE Description	DB9 Male pin	DCE Description	DB9 Female pin
Transmit Data	3	Receive Data	3
Receive Data	2	Transmit Data	2
Signal Ground	5	Signal Ground	5

*Table 6.2 3 Wire Cross Cable*

DTE Description	DB9 Male pin	DTE Description	DB9 Male pin
Transmit Data	3	Receive Data	2
Receive Data	2	Transmit	3
Signal Ground	5	Signal Ground	5

The device can be forced as the master coordinator from the dipswitch. The key, that is used when an endpoint connection is established, is selected from the dipswitch. The device has seven external switches:

- Three On-Off toggle switches for Control point selection.
- Two Push to make switches for add and remove endpoint connections.
- Two On-Off toggle switches for Input control points.

The selection switches state is shown in Table 6.3.

*Table 6.3 Control point selection*

Description	Switch 1	Switch 1	Switch 3
Light Switch	Off	Off	On
Door Sensor	Off	On	Off
Dimmer Switch	Off	On	On
Light	On	Off	Off
Siren	On	Off	On

The controller has 5 LED's. They are as follows:

- Blue: Master coordinator
- Yellow: Active
- Red: Busy
- Green: Output control endpoint #1
- Green: Output control endpoint #2

The activity light will flash every second to show that the microcontroller is operational. If a add connection is unsuccessful, it will flash every 100 milliseconds for three seconds. The busy LED indicates that both the transmit buffers are occupied.

## **6.4 RFD21733 Wireless Transceiver**

### **6.4.1 Wireless circuit diagram**

The transceiver's MAX-232 was connected as a DCE device. The same transceiver board was used to connect to the controller board or to the PC's, simulating a device. The final device was built with the RFD21733 transceiver on the controller board without any MAX-232 IC's. The RFD21733 was soldered onto a daughterboard, because it has non-standard pin outs, which connect to Connector 2 and 3 as illustrated in the circuit diagram in Figure 6.17.

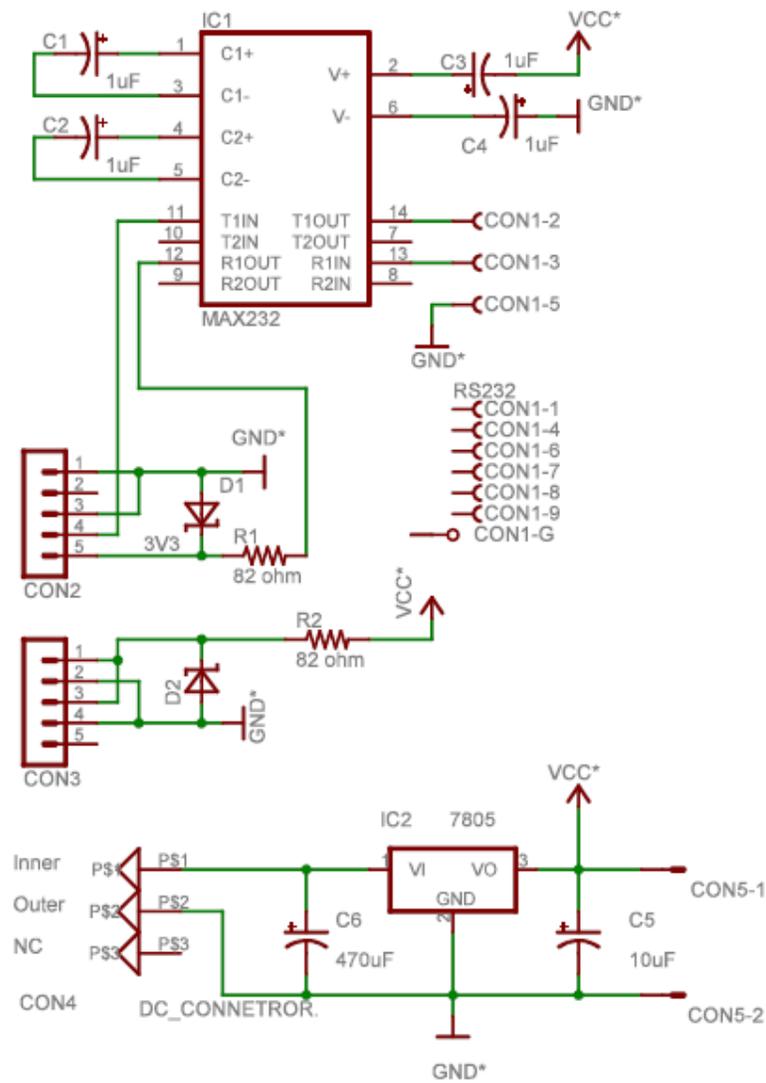


Figure 6.17 Wireless modem circuit diagram

## 6.4.2 Device and modem hardware

The Radio modem's hardware is shown Figure 6.18. Figure 6.19 depicts the bottom view of the modem hardware and the Radio transceiver daughterboard. In Figure 6.20 the hardware of the 'device' including the controller and wireless modems can be seen.

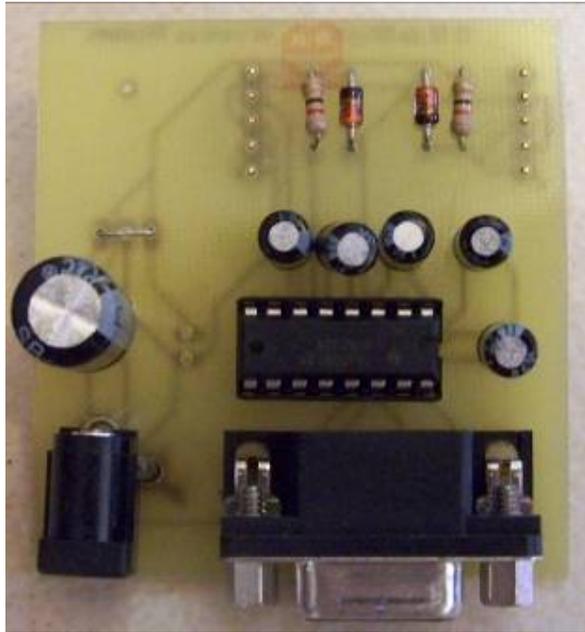


Figure 6.18 Radio modem top view

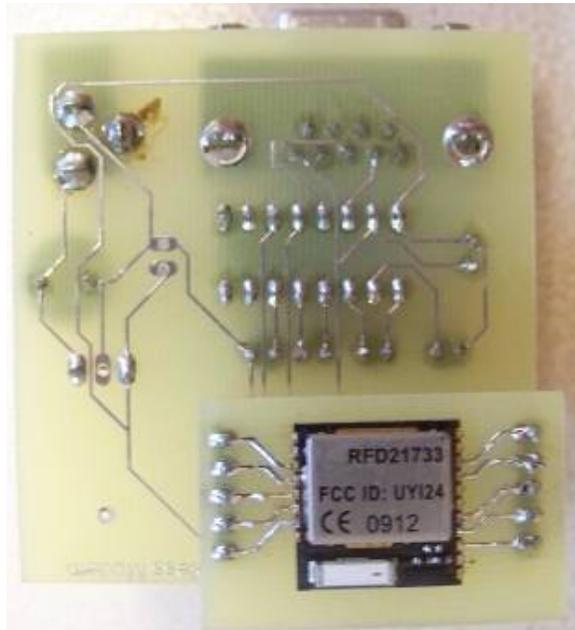


Figure 6.19 Radio modem bottom view with daughterboard

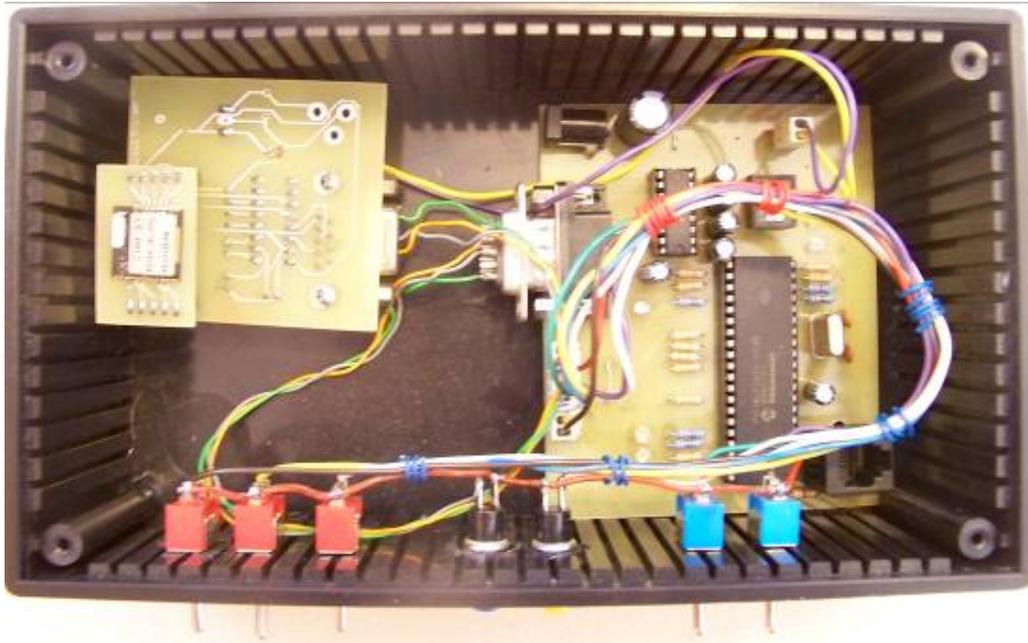


Figure 6.20 Device including controller and radio modem

### 6.4.3 Collision avoidance

Two devices/simulators with transceivers were tested without any noted problems. When the third device with transceiver was started all three devices started to lose connection with the other devices. The protocol was developed to disable transmission when it is receiving data. After investigating the simulator's received data, it was found that the data was a combination of two packets from two different devices as illustrated in Figure 6.21.



Figure 6.21 Packets received interlaced

The devices need to delay its transmission for the complete receiving frame when it senses that the channel is occupied. The line speed is 9600 bits per second. A data packet is the longest and is 34 bytes. Thus,

$$\begin{aligned} \text{Packet receiving time} &= 1 / (9600/8) \times 34 \\ &= 28.33 \text{ milliseconds} \end{aligned} \quad (6)$$

Initially the devices were configured to defer transmission for 30 milliseconds, but this was unsuccessful. The reason for this was that when the third device requested to join the WPAN, both receiving devices would wait 30 milliseconds and start to transmit an offer simultaneously. The requesting device would still receive both replies interlaced. The fixed time value was changed to vary randomly between 0 and 200 milliseconds. When the devices received the first byte from a packet, it would randomly defer transmission. The three devices would start to communicate, but the communication between the devices was unstable and communication was lost intermittently between the devices. The only explanation was that the devices would still send data occasionally at the same time.

It was decided to use the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) technique as described in the section about collisions in section 2.2.2 to control the collisions between the devices. The device will now send a 0xCC byte 20 microseconds before any transmission. Care must be taken that 0xCC is not used in any field of any frame. If any device received a 0xCC byte, it would reset its data received counter to 0. If the device does not receive any data in this time, it would wait another random period, between 0 and 100 milliseconds, before the packet is transmitted.

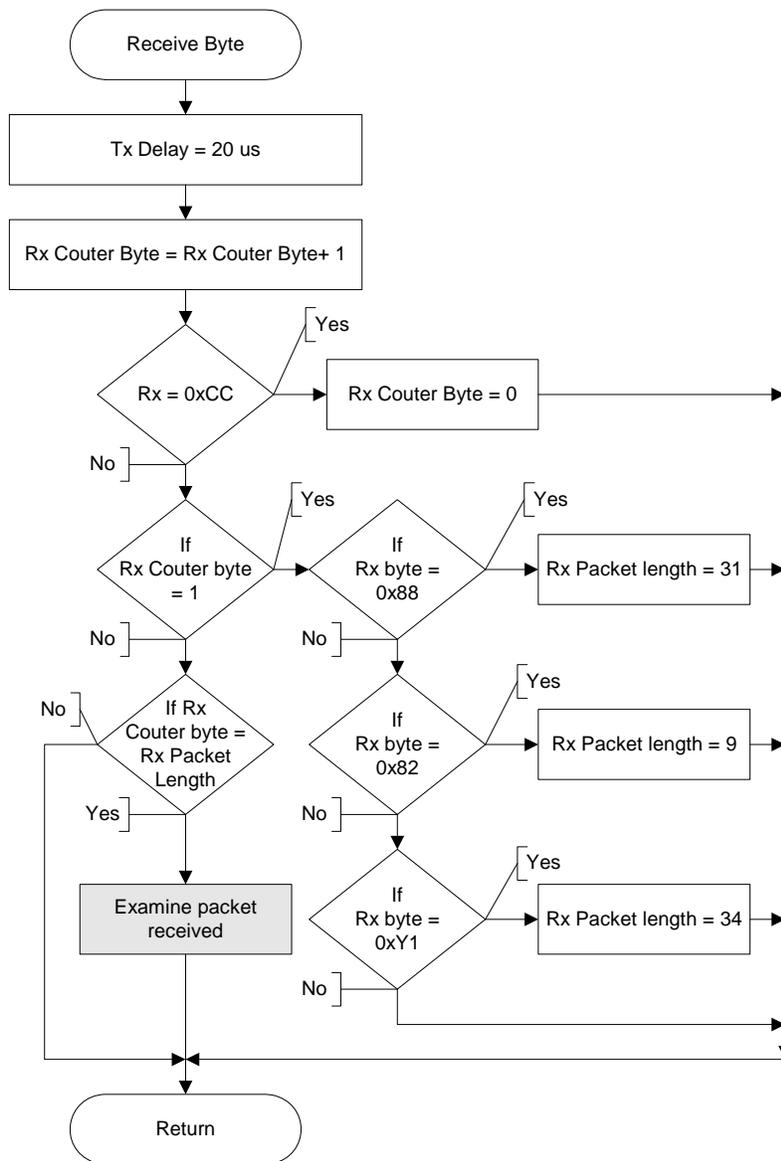


Figure 6.22 Collision receive byte

Figure 6.22 illustrates the process when the processor receives a byte via the USART. Firstly, the transmit delay timer is set to 20 microseconds. If 0xCC is received the byte counter is reset. This counter is used to determine when the end of a packet is received. When the counter is one, the algorithm will link the length of the frame to the first byte. Discovery frames start with a 0x88 and is

31 bytes, a Keepalive frame starts with 0x82 and is 9 bytes, and a Data frame starts with 0xY1 and is 34 bytes in length. The first nibble is the TTL field and varies between 0x1 and 0x8 for a data field. The reason for this is that it is the only frame that is routable. The complete frame is investigated after the associated number of bytes is received.

Figure 6.23 illustrates the process when the algorithm sends a frame to the USART. If the 20 us transmit timer is active and the first buffer is spare, the frame is stored in the first buffer. If the timer idles and the second buffer is empty, the frame is stored in the second buffer. A random timer for the second buffer is started and a RTS byte, 0xCC, is transmitted.

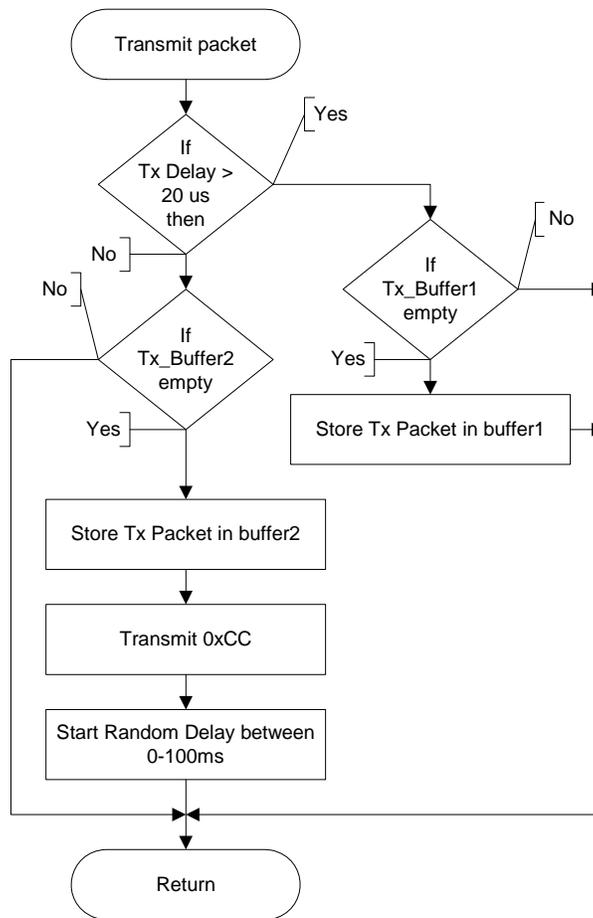


Figure 6.23 Collision transmit packet

Figure 6.24 illustrates when a collision related timer expires. A RTS byte, 0xCC, is transmitted when the transmit delay timer expires and the second buffer is occupied. The random delay timer is started last. When the second buffer becomes spare, the information from the first buffer is moved to the second buffer. The RTS byte is then transmitted and the random timer started. The content of Buffer 2 is transmitted when the random timer expires and the transmit delay timer idles. If the transmit delay timer is set, the content of the second buffer is delayed again for a random time.

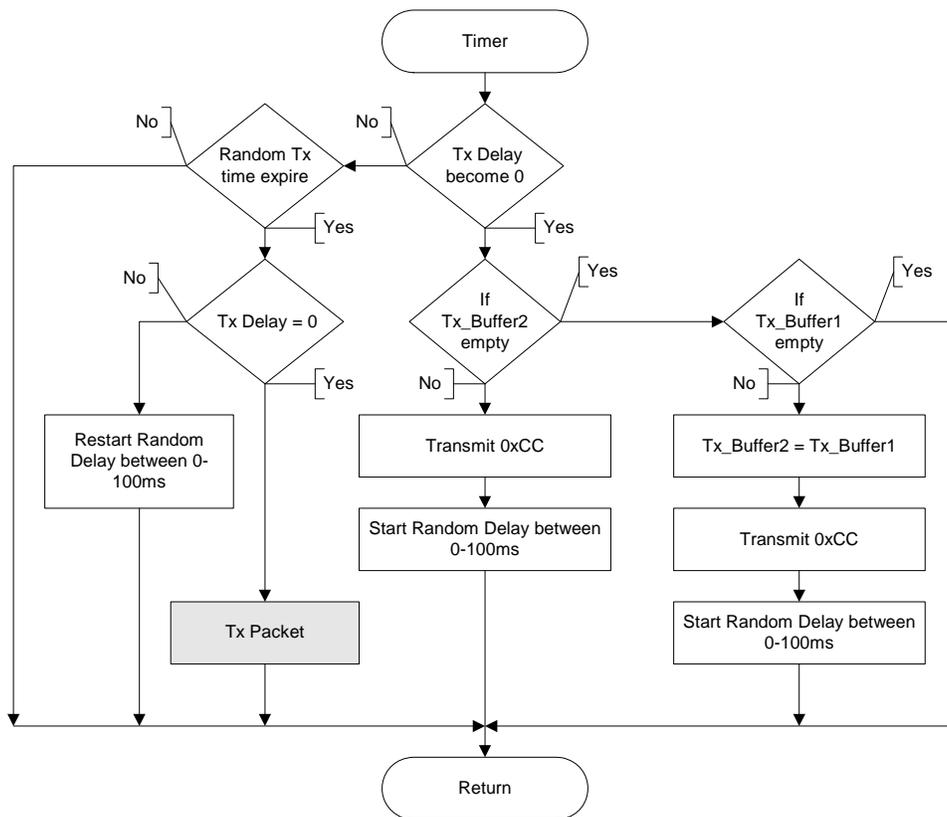


Figure 6.24 Collision timer

One of the biggest mistakes made within this project was to underestimate the influence of collisions in a wireless network.

## 6.5 Final test results

The final test was done with two personal computers and one device connected together via a wireless modem.

The goal of the test was to investigate how each command suite would react with an increase of devices and endpoints. A Window tab was designed for the simulator to count the different commands as illustrated in Figure 6.25

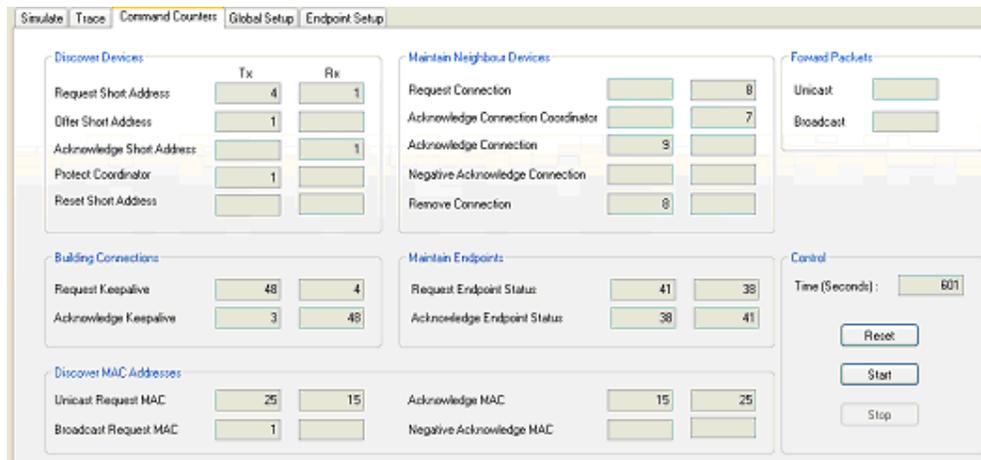


Figure 6.25 Screenshot of the Command Counter window

Each test was done the same. The procedure is as follows:

- Duration: 10 minutes
- Add connections: at 15 seconds
- Change endpoints status: at 2, 5 and 8 minutes
- Remove connections: at 9 minutes

The test started with one simulator with zero connected endpoints. With the second test, two endpoints were connected on a single device. The only traffic in both cases was when the device started and requested a short address as shown in Table 6.4.

*Table 6.4 Packet count with one device*

<b>Device</b>	1		1	
<b>Endpoints</b>	0		2	
<b>Commands</b>	<b>TX</b>	<b>RX</b>	<b>TX</b>	<b>RX</b>
Request Short Address	4		4	
Offer Short Address				
Acknowledge Offer				
Protect Coordinator	1		1	

A second simulator was added and the endpoint connections were increased in each test to a maximum of eight endpoint connections per simulator. The test results are shown in Table 6.5 and Table 6.6. The first device always started as the coordinator and the second PC simulator always received the short address 0x0100. The delay was minimum each time the endpoint status changed to reflect at the remote side. With eight and sixteen endpoints connected there were cases where the status change did not reflect immediately at the remote side. This is the reason for the broadcast resolve MAC in Table 6.6 in the column with 4 and 8 endpoint connections per device, highlighted in the circle.

Table 6.5 Packet count with two devices, Part 1

Device Endpoints Commands	1		2		1		2		1		2	
	TX	RX										
Request Short Address	2	2	2		4	2	2		4	2	2	
Offer Short Address	2			2	2			2	2			2
Acknowledge Offer		2	2			2	2			2	2	
Protect Coordinator	1				1				1			
Request Keepalive	4	46	46	3	3	49	49	3	48			48
Acknowledge Keepalive	46	4	3	46	42	3	3	42		48	48	
Unicast Request MAC									1			1
Broadcast Request MAC										1	1	
Acknowledge MAC												
Request Connection									2	2		
Ack. Connection Coordinator									2	2		
Acknowledge Connection									3			3

Table 6.6 Packet count with two devices, Part 2

Device Endpoints Commands	1		2		1		2		1		2	
	TX	RX										
Request Short Address	4	3	3		4	2	2		4	2	2	
Offer Short Address	2			2	2			2	2			2
Acknowledge Offer		2	2			2	2			2	2	
Protect Coordinator	1				1				1			
Request Keepalive	26	27	27	26	45	5	5	45	47	4	4	47
Acknowledge Keepalive	27	26	26	27	5	45	45	5	4	47	47	4
Unicast Request MAC	4			4	5	5	5	5	9	5	5	9
Broadcast Request MAC									1		1	1
Acknowledge MAC		4	4		5	5	5	5	5	9	9	5
Request Connection		1	1			2	2		4	4		
Ack. Connection Coordinator		1	1			2	2		3	4		
Acknowledge Connection	1			1	2			2	5			5
Remove Connection	1			1	2			2	4			4
Request Endpoint Status	9			9	10	11	11	10	16	17	17	16
Ack. Endpoint Status		9	9		11	10	10	11	17	16	16	17

Local			Remote		
Description	Select	Status	Remote Address	Description	Status
Door Sensor	<input type="radio"/>	255	4239563186718750/02/0100/87	Alarm	<input type="checkbox"/>
Alarm	<input type="radio"/>	<input type="checkbox"/>	4239563186718750/01/0100/5	Door Sensor	255
Light Switch	<input type="radio"/>	255	4239563186718750/04/0100/85	Light Outside	<input type="checkbox"/>
Light Outside	<input type="radio"/>	<input type="checkbox"/>	4239563186718750/03/0100/3	Light Switch 1	255
Gas Front Sensor	<input type="radio"/>	0	4239563186718750/07/0100/60	Sensor1	255
Alarm	<input type="radio"/>	<input type="checkbox"/>	4239563186718750/06/0100/77	Alarm	<input type="checkbox"/>
Sensor1	<input type="radio"/>	0	4239563186718750/03/0100/63	Dim Light	<input type="checkbox"/>
Dim Switch	<input type="radio"/>	50	4239563186718750/05/0100/1	Gas Front Senso	255
Irigation	<input type="radio"/>	<input type="checkbox"/>			
Spere	<input type="radio"/>	0			

Figure 6.26 Screenshot of eight endpoints connected

Figure 6.26 shows a screenshot of one of the simulators with eight endpoints connected to the other simulator.

Finally, two simulators and one device were used. With the first test, the first simulator becomes the coordinator and the second simulator did accepted short address 0x0100. When the devices started, it first did accept short address 0x0110. When it later requested a better short address, it did accept short address 0x0200.

With the second test, an endpoint was connected from the coordinator to the second simulator and an endpoint from the device to the second simulator. The short addresses were 0x0000, 0x0100 and 0x0200 for the whole duration.

With the third test, the short address of the third device started as 0x0110 and changed later to 0x0200. The endpoints connected learned the new address and still managed to control the remote endpoints. After the 10-minute test period, the third device and the first device, became coordinators. When the second device requested a better short address, it did get an offer from two different coordinators. It did request one of the coordinators to request a short address and the WPAN recovered automatically. The test result is shown in Table 6.7.

Table 6.7 Packet count with three devices, Part 1

Device	1			2			3					
	0		0	0		0	1		1	2		2
Endpoints	TX RX		TX RX		TX RX		TX RX		TX RX		TX RX	
Commands	TX RX		TX RX		TX RX		TX RX		TX RX		TX RX	
Request Short Address	3	7	2	5	2	7	2	5	4	8	2	6
Offer Short Address	7		5	2	7		5	2	8		6	3
Acknowledge Offer		3	2	3		3	2	3		3	2	1
Protect Coordinator	1				1				1		1	1
Request Keepalive	27	43	26	80	48	21	4	101	4	76	34	33
Acknowledge Keepalive	43	27	77	26	21	49	98	4	74	4	33	45
Unicast Request MAC					2	4	4	3	13			9
Broadcast Request MAC									4	3		8
Acknowledge MAC					4	2	3	4	1	15	12	
Request Connection						3	2			4	2	
Ack. Connection Coordinator						2	2			2	2	
Acknowledge Connection					4			5	7			2
Remove Connection					1			2	1			2
Request Endpoint Status						9	9	13	23			25
Ack. Endpoint Status					8		11	8		21	22	
Forward Unicast									14			
Forward Broadcast									1			

The fourth and fifth test results are shown in Table 6.8. In these tests the short address of the third device also changed between 0x0110 and 0x0200. As the endpoint connections increased, more scenarios occurred where there were a delay from when the endpoint input status changed until it reflects at the remote side.

If the developer needs faster response times, the interval between request statuses, request MAC and Broadcast MAC can be shortened. This will have an impact on the amount of connected endpoints hosted on a device.

Table 6.8 Packet count with three devices, Part 2

Device	1			2			3					
	4		4		4		7		7		4	
Endpoints	TX		RX		TX		RX		TX		RX	
Commands	TX		RX		TX		RX		TX		RX	
Request Short Address	2	6	1	5	4	4	1	3				
Offer Short Address	6		5	1	4		3	1				
Acknowledge Offer		4	1			2	1	2				
Protect Coordinator	1				1		1					
Request Keepalive	47	69	2	50	21	77	48	61				
Acknowledge Keepalive	67	48	47	2	75	21	61	47				
Unicast Request MAC	15	9	11	13	47	20	12	43				
Broadcast Request MAC	2	3		5	4	21		19				
Acknowledge MAC	10	18	17	11	37	53	51	12				
Request Connection		9	5			16	12					
Ack Connection Coordinator		7	4			12	10					
Acknowledge Connection	13			6	22			11				
Remove Connection	3	2	1	4	11	3	3	10				
Request Endpoint Status	20	23	18	21	54	27	18	55				
Ack Endpoint Status	20	18	20	19	23	35	39	17				
Forward Unicast	67				29		56					
Forward Broadcast							8					

These test results were used to calculate the tables and graphs shown below. The maximum bandwidth available in the test was 9600 bits per second and it is a hardware limitation of the transceiver.

The bits transmitted were calculated as follows:

$$\text{Bits transmitted} = (\text{Frames Transmitted} \times \text{suite frame length} \times 8) / 600 \quad (7)$$

The Frame count is multiplied with the frame length of the suite. The Discovery suite is 31 bytes, 9 bytes for Keepalives and 34 bytes for the data frames. To convert the bytes into bits, the count of frames transmitted is multiplied by 8. This result is then divided by 600, because all the tests were done over a 10-minute period, for an average bit rate per second.

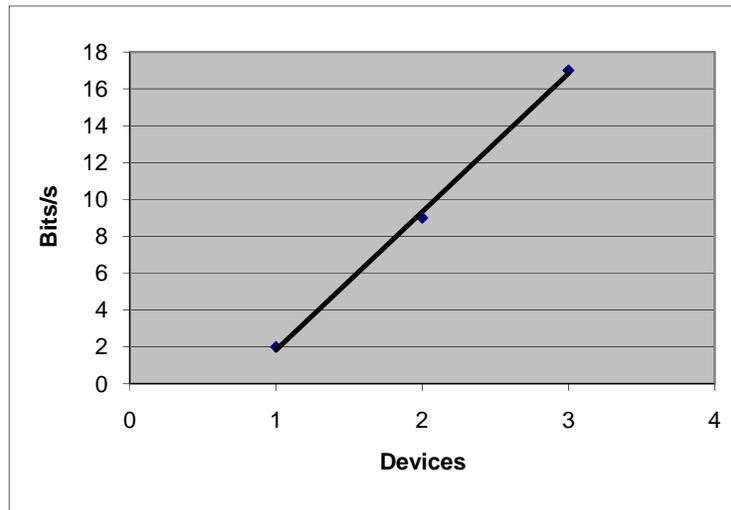
*Devices vs. bandwidth*

To calculate the minimum bandwidth required per device, only Discovery and Keepalive frames were taken into account for this calculation. These two command suites will be present when a device is part of a WPAN, with or without any endpoint connections.

Table 6.9 illustrates the relationship between the amount of devices and bits per second. It is important to note that there was almost no increase in Discovery or Keepalive frames with an increase in endpoint connections.

*Table 6.9 Relation between devices vs. bits per second*

Device	1		2				3					
Endpoints	0	2	0	1	2	4	8	0	1	2	4	7
Average bits/s per Endpoint	2	2	8	9	8	9	7	16	12	16	17	16
Maximum Bits/s per Device	2		9				17					



*Figure 6.27 Relation between devices vs. bits per second*

Figure 6.27 illustrates that the relationship between the bits per second, the amount of devices and the trend is a linear line. The linear lines formula can be calculated as follows:

$$(y_2 - y_1) = m (x_2 - x_1) + c$$

$$(17 - 2) = m (3 - 1) \text{ where } c = 0 \text{ for calculating the gradient}$$

$$\text{Thus, } m = 7.5 \tag{8}$$

Where

$$y = mx + c$$

Substitute y and x

$$2 = 7.5 (1) + c$$

$$c = -5.5 \tag{9}$$

Final result

$$y = 7.5x - 5.5 \quad \text{where } y = \text{bits/s and } x = \text{number of devices} \tag{10}$$

The maximum number of devices, without connections to their endpoints, with a 9600 bits per second wireless network, can be calculated as follows:

$$\begin{aligned} \text{Maximum devices} &= (9600 + 5.5) / 7.5 \\ &= 1280.733 \end{aligned} \tag{11}$$

*Connection vs. bandwidth*

In this calculation, only Data frames were taken into account to determine the relationship between bandwidth and endpoint connections.

Table 6.10 illustrates the summarised test results for the number of bits per second transmitted versus the amount of connections per number of devices.

Table 6.10 Relation between endpoint connections vs. bits per second

Device	1		2				3					
Endpoints	0	2	0	1	2	4	8	0	1	2	4	7
Average bits/s per endpoint	0	0	0	5	10	17	40	0	8	26	62	94

Figure 6.28 shows the above-mentioned information in graph form.

Table 6.10 illustrates the relationship between bits per second and endpoints used with different amount of devices. With one, two or three devices the trends were linear lines.

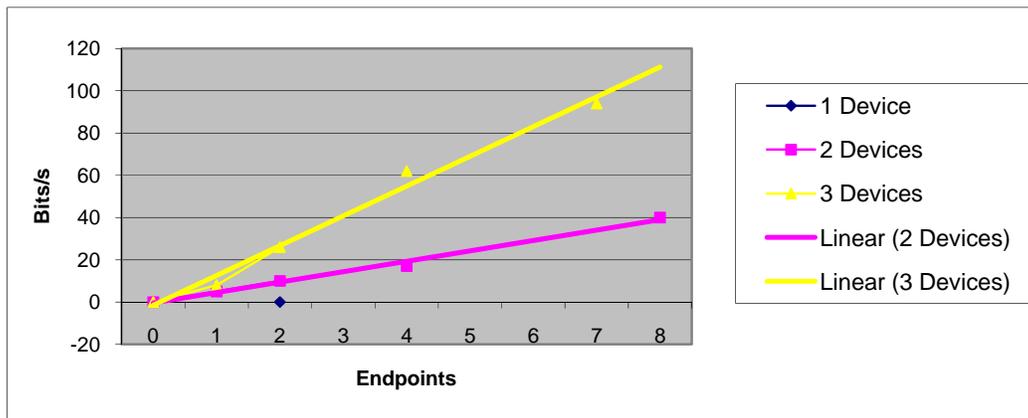


Figure 6.28 Relation between endpoint connections vs. bits per second

The relationship between bandwidth, device and endpoints can be calculated as follows:

Two devices linear line formula is:

$$(y_2 - y_1) = m (x_2 - x_1) + c$$

$$(40 - 10) = m (8 - 2) \text{ where } c = 0 \text{ for calculating the gradient}$$

$$\text{Thus, } m = 5.$$

Where

$$y = mx + c$$

Substitute y and x

$$40 = 5(8) + c$$

$$c = 0$$

Final result:

$$y = 5x \quad \text{where } y = \text{bits/s and } x = \text{number of endpoints} \quad (12)$$

Three devices linear is formula is:

$$(y_2 - y_1) = m(x_2 - x_1) + c$$

$$(94 - 8) = m(7 - 1) \quad \text{where } c = 0 \text{ for calculating the gradient}$$

$$\text{Thus } m = 14.333.$$

Final result:

$$y = 14.333x \quad \text{where } y = \text{bits/s and } x = \text{number of endpoints} \quad (13)$$

The gradient (m) changes for each line and needs to be calculated to determine  $n$  devices linear line formula. The  $m_n$  variable can be calculated for each device line as follows:

$$m_0 = 0, m_1 = 5 \quad \text{and } m_2 = 14.333 \quad \text{where } x = 1.$$

$$\Delta m_1 \text{ \& } m_0 = 5 \quad (14)$$

$$\Delta m_2 \text{ \& } m_1 = 9.333 \quad (15)$$

$$\Delta (15) \text{ \& } (14) = 4.333 \quad (16)$$

The Delta in gradient  $m$  between  $\Delta m_1 \text{ \& } m_0$  and  $\Delta m_2 \text{ \& } m_1$  is 4.333. To determine the following  $\Delta$  of  $\Delta$  values, a forecast was made that  $\Delta$  of  $\Delta m_n$  and  $m_{n-1}$  will be 4.333. Table 6.11 illustrates the Delta in gradient  $m$  between two linear lines.

Table 6.11 Delta gradient between different lines

Delta $m_2$ & $m_1$	5
Delta $m_3$ & $m_2$	9.33
Delta $m_4$ & $m_3$	13.667
Delta $m_5$ & $m_4$	17.996
Delta $m_6$ & $m_5$	22.329
Delta $m_7$ & $m_6$	26.662
Delta $m_8$ & $m_7$	30.995
Delta $m_9$ & $m_8$	35.328
Delta $m_{10}$ & $m_9$	39.661

The gradient  $m$  for each line, with one endpoint connection, can be calculated as follows:

$$m(n) = m(n-1) + \Delta m_{n-3} \& m_{n-2} \quad (17)$$

Table 6.12 and Figure 6.29 illustrates the gradient  $m$  value for each linear line in Figure 6.28 with an increase in devices.

Table 6.12 Bandwidth gradient increase with increase in devices

$n$	1	2	3	4	5	6	7	8	9	10
$m$	0	5	14.333	28	46	68.325	94.987	126	161.31	201

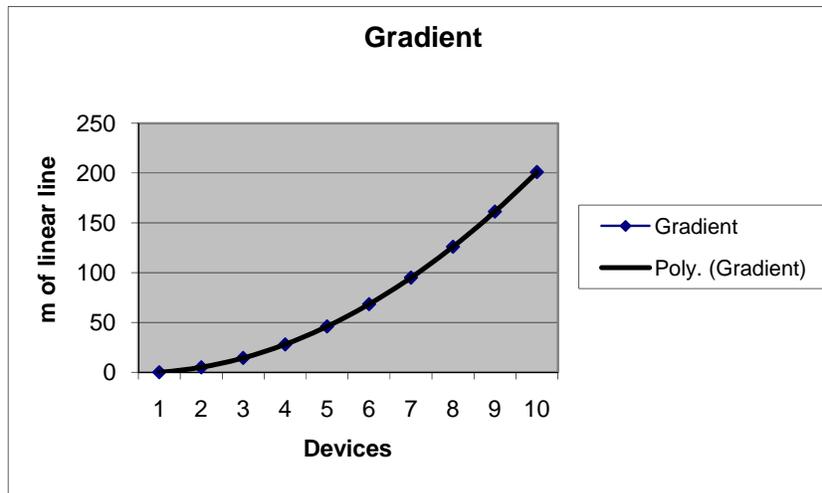


Figure 6.29 Relation between linear line gradient from Figure 6.28 and devices

From Figure 6.29 it seems that the relationship is parabola and the formula for a parabola is:

$$y = ax^2 + bx + c \quad (18)$$

The formula for the graph in Figure 6.29 can be calculated as follows:

Take three (x,y) point from Table 6.12 , where x = n and y = m and solve equation (18).

For x = 1 and y = 0

$$y = ax^2 + bx + c$$

$$0 = a.1 + b.1 + c$$

$$a = -(b+c) \quad (19)$$

For x = 4 and y = 28

$$y = ax^2 + bx + c$$

$$28 = 16a + 4b + c \quad \text{substitute a with (19)}$$

$$28 = 16 * -(b + c) + 4b + c$$

$$28 = 12b - 15c$$

$$c = (-12b - 28) / 15 \quad (20)$$

For x = 8 and y = 126

$$y = ax^2 + bx + c$$

$$126 = 64a + 8b + c \quad \text{substitute a with (19)}$$

$$126 = 64 * -(b + c) + 8b + c$$

$$126 = -56b - 63c \quad \text{substitute c with (20)}$$

$$126 = -56b - 63(-12b - 28)/15$$

$$126 = -56b - 50.4b + 117.6$$

$$b = -1.5 \quad (21)$$

$$c = (-12b - 28) / 15 \quad \text{from (20)}$$

$$c = ((-12 * -1.5) - 28) / 15 \quad \text{b substituted with (21)}$$

$$c = -0.667 \quad (22)$$

$$a = -(b + c) \quad \text{from (19)}$$

$$a = -(-1.5 + (-0.667)) \quad \text{substitute b and c with (21) and (22) respectively}$$

$$a = 2.167 \quad (23)$$

The gradient m for the linear lines in Figure 6.28 is:

$$m = 2.167n^2 - 1.5n - 0.667 \quad \text{where n = number of devices} \quad (24)$$

The total bandwidth required can be calculated as follows, where y = bandwidth, n = number of devices and x = average number of endpoints per device.

Bandwidth = (Bandwidth for Discovery and Keepalive) per device + (Bandwidth per endpoint) per device.

$$y = \text{formula (10)} + \text{formula(12) where m = formula (24)}$$

$$y = 7.5n - 5.5 + x(2.167n^2 + -1.5n - 0.667) \quad (25)$$

The maximum devices with a RFD21733 transceiver, 9600 bits/s transceiver, with one endpoint connection per device can be calculated as follows:

$$9600 = 7.5n - 5.5 + 2.167n^2 + -1.5n - 0.667$$

$$9600 = 2.167 n^2 + 6n$$

$$n \approx 65 \quad (26)$$

The maximum device will decrease if the number of endpoints per device increases.

## **6.6 Security of data**

No encryption of the data was incorporated on the microcontroller. The first reason: this would increase the risk that the microcontroller would be too busy to process the Receive data interrupt routine. The second reason: there are many IEEE 802.15.4 compliant transmitters that have built-in AES encryption available on the market today. Developing an encryption protocol is not deemed necessary at this stage for research purposes. A transceiver that is available and has built-in AES encryption is the Microchip's MRF24J40 transceiver. Its features are illustrated in Appendix A.

## **6.7 Conclusion**

In this chapter the hardware including the firmware was developed for a PIC18LF462 microcontroller. Wireless modems were built to connect to the PC simulators. The Resolve MAC process was changed to decrease the amount of data transmitted. The Add connection process was changed to make it more reliable. The biggest problems experienced were buffer overrun errors at the USART and to manage collisions.

## Chapter 7

### 7 Conclusion

In this dissertation the main objective of this research was the development of an alternative protocol for controlling multiple mobile control points to be controlled from one another. This protocol is based on a combination of varied solutions from several communication protocols like 802.15.4, Zigbee, MiWi™, UDP/IP.

A simulation and software system was developed where a device can host multiple control endpoints.

The protocol developed will let the device automatically negotiate which device becomes the master controller in the WPAN. The short address structure developed, incorporating automated neighbourhood relationship maintenance, allows the protocol to support endpoints on mobile devices. This short address structure differs from the 10-bit network or short address structure used in Zigbee and MiWi where it is an implementation reliant protocol.

Utilising this protocol, any similar endpoint that is hosted on a device, can be linked together and by each other. The protocol utilises a key-in feature where the add connection process allows the coordinator to administer multiple non-established connections in progress.

Evaluation and testing of this protocol deemed it necessary to develop a simulator where the protocol was simulated with UDP/IP.

A microcontroller hardware device was developed that includes single channel wireless transceivers. Collision avoidance allows the protocol to minimise collisions and the endpoints could be controlled even if the short addresses of the devices changed.

Most of the energy consumption in sensor networks takes place during transmission between sensor units. This research does not take energy consumption into account and can be developed and implemented for evaluation in future research.

## References

1. Forsberg B. **Wireless devices can act as remote controls for household or business**. San Francisco Chronicle, 8 January 2008.
2. Eady, F. **Implementing 802.15.4 with Microcontrollers**. Oxford, Elsevier, pp. x, 4, 6-7, 9-10, 2007.
3. LAN/MAN Standards Committee of the IEEE Computer Society. **Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)**. New York, pp. iii, 1, 13-18, 21-24 28, 67, 311, 2006.
4. Wikipedia. 2008. Zigbee [online], **Zigbee**. Available from <http://en.wikipedia.org/wiki/Zigbee>. [Accessed on 25 March 2008].
5. Anderson, D. **Networking Essentials**. Berkeley, Osborne McGraw-Hill, pp. 22-24, 38-43, 45, 53, 57, 73, 97, 1998.
6. Wikipedia, 2008, OSI Model [online], **OSI Model**, Available from [wiki/OSI\\_Model](http://en.wikipedia.org/wiki/OSI_Model), pp. 1- 4, 2009 [Accessed on 4 November 2008].
7. Lammle, T. **Cisco Certified Network Associate Study Guide**. Alameda, Sybex, pp. 26, 34, 42, 88, 1996.
8. Young, P.H. **Electronic Communication Techniques**. 3<sup>rd</sup> ed. New York, Macmillan Publishing Company, pp. 538-539, 1994.
9. Microchip. 2002. Wireless – Application – PIC16 – AN821 [online] **Advance Encryption Standard using the PIC16XXX**. Available from <http://ww1.microchip.com/downloads/en/AppNotes/00821a.pdf>, pp. 5-6. [Accessed on 4 September 2009].
10. Wikipedia. 2009. Cyclic redundancy check [online], **Cyclic redundancy check**, Available from [http://en.wikipedia.org/wiki/Cyclic\\_redundancy\\_check](http://en.wikipedia.org/wiki/Cyclic_redundancy_check). [Accessed on 3 December 2009].

11. Harold, F. **Information Security Management Handbook**. 5<sup>th</sup> ed. Boca Raton, CRC Press, pp. 1336-1339, 2004.
12. Microchip. 2007. MiWi Wireless – MiWi – MiWi – AN1066 [online] **MiWi Wireless Networking Protocol Stack**. Available from <http://ww1.microchip.com/downloads/en/AppNotes/01066a.pdf>, pp. 1-5, 7-9, 13. [Accessed on 23 March 2008].
13. Zigbee Alliance. 2009. Downloads – Latest Zigbee specification including the pro feature set [online], **Zigbee Specification**, Available from <http://www.zigbee.org/Products/TechnicalDocumentsDownload/tabid/237/Default.aspx>, pp. 1-2, 17-19, 21, 77, 82-83. [Accessed on 15 February 2008].
14. Anderson, D. **Microsoft TCP/IP on Windows NT4.0**. Berkely, Osborne McGraw-Hill, p. 171, 1998.
15. Lammle, T. **Cisco Certified Network Associate**. Alameda, Sybex Network Press, p. 689, 1999.
16. RF Design. 2010. 2.4 GHz Wireless Products – Data transceiver modules – RF Digital Tanceiver Module with RFD8 protocol Chip Antenna [online], **Complete 2.4 GHz RF Tranceover Module with Built\_In RFD8 Application Protocol**, Available from <http://rfdesign.co.za>, pp. 2, 13. [Accessed on 10 February 2010].
17. Microchip. 2008. 8bit – Pic18 Family – Pic18F4620 – PIC18F2525/2620/4525/4620 Data Sheet [online], **PIC18F2525/2620/4525/4620 Data Sheet**, Available from <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010304>, pp. 57, 121, 2008. [Accessed on 12 April 2010].
18. Microchip. 2005. Support – Technical support – 24/7 Tech support – Search – OERR [online], **What causes USART framing (RCSTA<FERR>) or overrun (RCSTA<OERR>) errors**, Available from <http://support.microchip.com/scripts/slxweb.dll/external?name=webticketcust>. [Accessed on 10 June 2010].

19. Floyd, T. **Digital Fundamentals**. 3<sup>rd</sup> ed. Ohio, Charles E Merrill Publishing Company, p. 289, 1986.

## Appendix A: MRF2470 Features



# MRF24J40

---

## IEEE 802.15.4™ 2.4 GHz RF Transceiver

---

### Devices Included:

- MRF24J40

### Features:

- Complete IEEE 802.15.4 Specification Compliant
- Supports MiWi™, ZigBee™ and Proprietary Protocols
- Simple, 4-Wire SPI Interface
- Integrated 20 MHz and 32.768 kHz Oscillator Drive
- 20 MHz Reference Clock Output:
  - Available to drive microcontroller oscillator
- Supports Power-Saving mode
- Low-Current Consumption, Typical 18 mA in RX mode and 22 mA in TX mode
- Typical 2  $\mu$ A Sleep mode
- Small, 40-Pin Leadless QFN 6x6 mm<sup>2</sup> Package

### RF/Analog Features:

- ISM Band 2.405-2.48 GHz Operation
- -91 dBm Typical Sensitivity and +5 dBm Maximum Input Level
- +0 dBm Typical Output Power and 38.75 dB TX Power Control Range
- Differential RF Input/Output and Integrated TX/RX Switch
- Integrated Low Phase Noise VCO, Frequency Synthesizer and PLL Loop Filter
- Digital VCO and Filter Calibration
- Integrated RSSI ADC and I/Q DACs
- Integrated LDO
- High Receiver and RSSI Dynamic Range

### MAC/Baseband Features:

- Hardware CSMA-CA Mechanism, Automatic ACK Response and FCS Check
- Independent Beacon, Transmit and GTS FIFO
- Hardware Security Engine (AES-128) with CTR, CCM and CBC-MAC modes
- Supports all CCA modes and RSS/LQI
- Automatic Packet Retransmit Capability
- Supports In-Line or Stand-Alone modes for both Encryption and Decryption