The Design of a Low Cost Ad-hoc Network for Short Distance Data Acquisition

by

Cornelius Marais Rossouw

Thesis presented in partial fulfillment of the requirements for the degree
Master of Science in Electronic Engineering

at Stellenbosch University

Supervisor:
Dr. R. Wolhuter

December 2008
Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly stated otherwise) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: November 27, 2008

Copyright © 2008 Stellenbosch University
All rights reserved
Abstract

In this thesis the design of a low-cost ad hoc network for short distance data acquisition applications with low data arrival intervals will be presented. The focus is on cost reduction by replacing the traditional high power radios with low-power RF transceivers. The conventional way of using multiple stationary repeater towers (depending on the network) is also replaced by using an ad hoc configuration, where each individual station also serves as a repeater station to adjacent stations. This approach reduces network design time enormously, seeing that the network is able to configure itself. By using this auto-routing multi-hop approach, data acquisition points are no longer restricted to the reception areas of base stations.

A CSMA contention protocol is used for the data communication. Current models used to model this protocol are dependent on various assumptions. In the research reported in this thesis, a statistical study of the collision probability is performed and the results used to expand the current CSMA models. Inter-dependent characteristics of this model are also further enhanced to provide a more realistic model. A simulink model of the particular CSMA protocol is also designed. Both the mathematical- and the simulink models provide relatively good predictions when compared to actual measured results.
Opsomming

Hierdie tesis handel oor die ontwerp en ontwikkeling van ’n lae koste ad hoc netwerk vir kort afstand moniteringstoeplaasings. Daar word veral gefokus op koste besparing deur die tradisionele hoë drywing senders te vervang met lae drywing senders en ontvangers. Die gebruik van stationêre seinherhalers word vervang deur die gebruik van ’n ad hoc protokol, waarvolgens elke stasie optree as beide ’n moniteringstasie en ’n herhaler vir buurstasies. Hierdie benadering verkort die ontwerpstye van ’n netwerk aansienlik, aangesien die stasies self hul roetes en konneksies bepaal. Die gebruik van automatisering padverkenning en multi-hop implementering, veroorsaak dat monitering nie meer beperk word tot die ontvangsareas van basisstasies nie.

’n CSMA kontensie protokol word gebruik vir data kommunikasie in hierdie geval. Huidige wiskundige modelle van hierdie protokol word gebaseer op baie aannames. ’n Statistiese studie van die waarskynlikheid van botsings tussen pakkies word in hierdie tesis bespreek. Die bevindinge van hierdie studie word toegevoeg tot die reeds bestaande model om ’n meer uitgebreide resultaat te lewer. Die afhanklikheid tussen die aantal botsings en die netwerk verkeersdigtheid word ook rekursief ge-analiseer om ’n meer realistiese resultaat te lewer. Die wiskundige model word vergesel deur ’n simulink model wat ’n meer direkte benadering tot analise volg. Beide hierdie modelle lever redelik akkurate resultate in vergelyking met werklik gemete waardes.
Acknowledgements

I would really like to thank the following people, without whom this thesis would never have been completed.

• My supervisor Dr. R Wolhuter for his guidance and insight into problems
• My family for their encouragement and support
• Maritza for her support, patience, motivation and understanding
• Mr. J Arendse and Mr. W Croucamp for the soldering of SMD chips and boards
• God
Contents

Declaration i

Abstract ii

Opsomming iii

Acknowledgements iv

Contents xi

List of Figures xviii

List of Tables xx

Nomenclature xxi

1 Introduction 1

1.1 Introduction and Background ........................................ 1

1.1.1 Reducing cost .................................................... 2

1.1.2 Feasibility study .................................................. 3

1.1.3 Prediction model .................................................. 3

1.1.4 Summary and Contributions ..................................... 3

1.2 Overview of the thesis ................................................. 4

2 Network Design 7
CONTENTS

2.1 Basic propagation principles .................................................. 7
  2.1.1 Free space propagation .................................................. 7
  2.1.2 Line of sight (LOS) links ................................................. 8
  2.1.3 Non-line of sight (NLOS) links ....................................... 12
  2.1.4 Software used for propagation predictions ......................... 13
2.2 Network design ................................................................. 13
  2.2.1 Software based design of optimal radio links ..................... 15
2.3 Summary ......................................................................... 29

3 Communication Strategy ......................................................... 31
  3.1 Physical layer ................................................................. 32
    3.1.1 Hardware ................................................................. 32
    3.1.2 Modulation .............................................................. 33
  3.2 Data link layer ............................................................... 33
    3.2.1 Error control ........................................................... 33
    3.2.2 Automatic repeat-request (ARQ) ................................... 34
    3.2.3 Forward error correction (FEC) .................................. 36
    3.2.4 Implemented error control ........................................ 37
    3.2.5 Basic transmission protocol ...................................... 38
  3.3 Network layer ............................................................... 39
    3.3.1 Routing ................................................................. 39
    3.3.2 Addressing ............................................................. 42
    3.3.3 Packet header construction ....................................... 44
  3.4 Transport and session layer ............................................... 45
  3.5 Presentation layer .......................................................... 45
    3.5.1 RF command set ...................................................... 45
3.6 Application layer ................................................................. 46
3.7 Scanning procedure .............................................................. 47
  3.7.1 Child list ................................................................. 47
  3.7.2 Scanning for orphans .................................................... 47
  3.7.3 Confirming old children ............................................... 49
  3.7.4 Searching for grandchildren ......................................... 49
  3.7.5 Complete scan sequence ............................................. 51
3.8 Summary ................................................................. 51

4 Hardware Design ............................................................ 53
  4.1 Component selection ..................................................... 54
    4.1.1 RF link ............................................................. 54
    4.1.2 Ethernet interface .................................................. 56
    4.1.3 Microcontroller ..................................................... 57
    4.1.4 Serial interface ..................................................... 58
    4.1.5 Power supply ....................................................... 58
  4.2 Circuit layout ............................................................. 58
    4.2.1 Connection of the nRF905 RF transceiver ....................... 58
    4.2.2 Connection of the ENC28J60 ethernet controller ............... 60
    4.2.3 Connection of the MAX3232 serial transceiver ................. 61
    4.2.4 Connection of the ADC port .................................... 61
    4.2.5 Connecting the power supply .................................... 61
    4.2.6 External connections of the microcontroller ................... 62
  4.3 PCB design and manufacturing ......................................... 63
  4.4 Cost ................................................................. 64

5 Embedded PIC software ....................................................... 70
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.2 ADC settings</td>
<td>93</td>
</tr>
<tr>
<td>6.3 Network maintenance</td>
<td>94</td>
</tr>
<tr>
<td>6.4 Data logging</td>
<td>95</td>
</tr>
<tr>
<td>6.5 Performance analysis</td>
<td>100</td>
</tr>
<tr>
<td>6.6 Error handling</td>
<td>102</td>
</tr>
<tr>
<td>6.7 Summary</td>
<td>103</td>
</tr>
<tr>
<td><strong>7  Performance Predictions</strong></td>
<td><strong>104</strong></td>
</tr>
<tr>
<td>7.1 Introduction</td>
<td>104</td>
</tr>
<tr>
<td>7.2 Queueing Theory</td>
<td>105</td>
</tr>
<tr>
<td>7.2.1 Background</td>
<td>105</td>
</tr>
<tr>
<td>7.2.2 Basic Queueing Principles</td>
<td>105</td>
</tr>
<tr>
<td>7.2.3 Modeling the Queueing System</td>
<td>107</td>
</tr>
<tr>
<td>7.3 Round-Robin Polling (RRP)</td>
<td>108</td>
</tr>
<tr>
<td>7.3.1 Background</td>
<td>108</td>
</tr>
<tr>
<td>7.3.2 Conventional Method</td>
<td>108</td>
</tr>
<tr>
<td>7.3.3 Queueing Approach</td>
<td>112</td>
</tr>
<tr>
<td>7.4 CSMA</td>
<td>118</td>
</tr>
<tr>
<td>7.4.1 Background</td>
<td>118</td>
</tr>
<tr>
<td>7.4.2 Collision probabilities</td>
<td>120</td>
</tr>
<tr>
<td>7.4.3 Setting up the Model</td>
<td>128</td>
</tr>
<tr>
<td>7.4.4 Extending the model to include collisions</td>
<td>137</td>
</tr>
<tr>
<td>7.4.5 Solving CSMA with collisions by adding a collision state</td>
<td>142</td>
</tr>
<tr>
<td>7.5 Simulink Model</td>
<td>143</td>
</tr>
<tr>
<td>7.5.1 Basic design of the RSB block</td>
<td>145</td>
</tr>
<tr>
<td>7.5.2 Design of the communication channel</td>
<td>152</td>
</tr>
</tbody>
</table>
CONTENTS

7.5.3 Design of the BS ................................................. 154
7.5.4 Network and simulation settings .................................. 154
7.5.5 Simulations and results ........................................... 156
7.6 Comparison .......................................................... 159
7.7 Summary ............................................................. 163

8 Measurements and Results ...................................... 165
  8.1 RF measurements ................................................... 165
  8.2 Latency measurements .............................................. 166
  8.3 Comparison between the simulated and measured results ............... 173
  8.4 Conclusion ........................................................ 179

9 Summary and Conclusion ..................................... 182
  9.1 Summary .......................................................... 182
  9.2 Conclusion ....................................................... 183
  9.3 Recommendations .................................................. 183

A Hardware design Information ................................ 186
  A.1 Design information on the nRF905 transceiver ....................... 186
  A.2 Design information on the PIC24FJ128GA006 ....................... 191
  A.3 Design information on the LM1117 voltage regulator ............... 194
  A.4 Design information on the MAX3232 serial transceiver ............. 194

B Multimedia guide ............................................. 195
  B.1 Root directory .................................................... 195
  B.2 Datasheets folder .................................................. 195
  B.3 Flowcharts and pictures from the report folder ..................... 196
  B.4 MATLAB code folder .............................................. 196
CONTENTS

B.5  Measured results folder ................................................................. 196
B.6  Other publications folder ............................................................... 196
B.7  PCB design folder ........................................................................... 196
B.8  Programming folder ......................................................................... 197
B.9  Software folder ................................................................................ 197

List of References .............................................................................. 199
# List of Figures

1.1 Supplied map of the area .............................................. 6

2.1 Fresnel Zone for a radio link (image taken from [McL01]) ............... 10
2.2 Fresnel Zone geometry based on knife edge diffraction(image taken from [McL01]) 10
2.3 Diffraction of a rounded obstacle (image taken from [McL01]) ............. 11
2.4 Basic flowchart of the MATLAB network link design ......................... 17
2.5 Flowchart of the link analysis procedure between two stations ............ 19
2.6 Example of a profile schematic for a link between two WiMax stations .... 20
2.7 Final links plot .................................................................. 22
2.8 UHF measured results taken from [Cou03] ...................................... 23
2.9 UHF predicted values taken from [Cou03] ...................................... 24
2.10 Screencapture of the network parameters and systems tabs ............... 24
2.11 Results for test 1a ................................................................ 26
2.12 Results for test 1b ................................................................ 27
2.13 Results for test 2a ................................................................ 28
2.14 Results for test 2c ................................................................ 29
2.15 A 3D representation of the results obtained from test 2c ............... 30
3.1 Schematic representation of the 7 layer OSI model .............................. 32
3.2 Manchester encoding .................................................................. 33
3.3 Schematic representation of the pure stop and wait protocol .............. 35
3.4 Block diagram of the acknowledge system ............................ 39
3.5 Example network layout of an ad-hoc network ....................... 41
3.6 Packet header layout ................................................ 45
3.7 Layout of the child entry in the child list .......................... 47
3.8 The basic scan procedure .......................................... 48
3.9 Flowchart of the complete scan sequence ........................... 52

4.1 RSB hardware block diagram ........................................ 53
4.2 APB hardware block diagram ....................................... 54
4.3 Radio mobile screen capture .................................... 56
4.4 nRF905-C433 completed module .................................. 59
4.5 Typical circuit used to convert current to voltage for ADC measurements .... 62
4.6 Crystal circuit design ............................................. 63
4.7 Enabling the internal LDO regulator ................................ 63
4.8 ICSP connection taken from [Mic05] ............................... 63
4.9 RJ11 connection taken from [Mic05] ............................... 63
4.10 The initial RSB layout ............................................. 64
4.11 The initial APB layout ............................................. 65
4.12 The final RSB layout ............................................... 66
4.13 PCB Top Layer (Scale 1:1) ......................................... 67
4.14 PCB Silkscreen (Scale 1:1) ......................................... 67
4.15 PCB Bottom Layer (Scale 1:1) ...................................... 68
4.16 Photo of the RSB .................................................... 69

5.1 Tasks performed by the C30 compiler (Taken from [MPL05]) .................. 71
5.2 Flowchart of the main PIC program .................................. 72
5.3 Different interrupts .................................................. 73
LIST OF FIGURES

5.4 Flowchart of the DR interrupt ........................................ 74
5.5 Flowchart of the serial interrupt ..................................... 75
5.6 Flowchart of the RTCC interrupt ..................................... 77
5.7 Flowchart of the back-off interrupt ................................. 78
5.8 Flowchart of the service routine interrupt ......................... 79
5.9 Flowchart of the push button interrupt ............................. 79
5.10 Flowchart of the delay implemented using timer 1 ............... 80
5.11 Program memory organization taken from [Mic07] ............... 82
5.12 Flowchart of the RTSP procedure .................................. 82
5.13 Flowchart of the TX process ....................................... 85
5.14 Flowchart of the RX data process .................................. 86
5.15 Flowchart of the Scan_Network and Scan_REQ procedures ........ 87
5.16 Flowchart of the Scan_Received procedure ....................... 88
5.17 Flowchart of the Scan_GrandChildren procedure ................ 89
5.18 Flowchart of an extract from the service interval procedure, which influences the scan process ........................................ 90

6.1 A screen capture of the Main tab of the server GUI ................ 94
6.2 A screen capture of the ADC settings tab of the server GUI ........ 95
6.3 Flowchart of the children logging procedure ....................... 96
6.4 Screen capture taken of the Network layout tab .................... 97
6.5 Flowchart of the GUI startup (initialization) procedure .......... 98
6.6 Flowchart of the Adjust_ADC procedure ............................ 99
6.7 Flowchart of the data logging process ............................... 100
6.8 Screen shot of the GRAPH tab of the Logs tab ....................... 101
6.9 Flowcharts of all the different performance analysis logging procedures .......... 102
6.10 Screen shot of the GRAPHS tab of the Performance tab ........... 103
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Basic queue model</td>
<td>105</td>
</tr>
<tr>
<td>7.2</td>
<td>Multi-hop system</td>
<td>110</td>
</tr>
<tr>
<td>7.3</td>
<td>Average waiting time vs. number of stations using conventional RRP</td>
<td>112</td>
</tr>
<tr>
<td>7.4</td>
<td>Average waiting time vs. number of stations for different topologies using conventional RRP</td>
<td>113</td>
</tr>
<tr>
<td>7.5</td>
<td>Average wait time vs. noise for different topologies using conventional RRP</td>
<td>113</td>
</tr>
<tr>
<td>7.6</td>
<td>Round-Robin queue model</td>
<td>114</td>
</tr>
<tr>
<td>7.7</td>
<td>Effect of noise on the RRP queue model</td>
<td>117</td>
</tr>
<tr>
<td>7.8</td>
<td>Comparison between the conventional and the queue RRP model</td>
<td>117</td>
</tr>
<tr>
<td>7.9</td>
<td>Effect of noise and arrival rate on the RRP queue model</td>
<td>118</td>
</tr>
<tr>
<td>7.10</td>
<td>CSMA hidden node effect</td>
<td>119</td>
</tr>
<tr>
<td>7.11</td>
<td>Two circles intersecting: Worst case scenario</td>
<td>121</td>
</tr>
<tr>
<td>7.12</td>
<td>Three circles intersect: Worst case scenario for more level 3 stations than level 2 stations</td>
<td>122</td>
</tr>
<tr>
<td>7.13</td>
<td>Determining the intersection points of three circles (taken from [Few06])</td>
<td>123</td>
</tr>
<tr>
<td>7.14</td>
<td>Three circles intersect: Worst case scenario for more level 2 stations than level 3 stations</td>
<td>125</td>
</tr>
<tr>
<td>7.15</td>
<td>CSMA Jackson network model</td>
<td>129</td>
</tr>
<tr>
<td>7.16</td>
<td>Basic state transition model</td>
<td>130</td>
</tr>
<tr>
<td>7.17</td>
<td>Effect of noise on CSMA</td>
<td>134</td>
</tr>
<tr>
<td>7.18</td>
<td>Utilization for CSMA model</td>
<td>135</td>
</tr>
<tr>
<td>7.19</td>
<td>Time spent in each queue vs different noise levels</td>
<td>135</td>
</tr>
<tr>
<td>7.20</td>
<td>Queue lengths for different noise levels</td>
<td>136</td>
</tr>
<tr>
<td>7.21</td>
<td>Effect multi-hop has on the system latency</td>
<td>137</td>
</tr>
<tr>
<td>7.22</td>
<td>Effect of hidden nodes on latency for CSMA networks</td>
<td>138</td>
</tr>
<tr>
<td>7.23</td>
<td>Effect of hidden nodes on utilization for CSMA networks</td>
<td>139</td>
</tr>
<tr>
<td>7.24</td>
<td>Effect of hidden terminals on queue lengths for CSMA networks</td>
<td>139</td>
</tr>
</tbody>
</table>
7.25 Effect of hidden nodes on time spent in each queue for CSMA networks .... 140
7.26 Effect of rise time on latency for CSMA networks .......................... 140
7.27 Effect of noise and collisions on latency for CSMA networks .............. 141
7.28 Effect of $\lambda$ on noise with CSMA networks .............................. 141
7.29 Extended state transition diagram for CSMA, modeling collisions by adding a collision state ................................. 142
7.30 Effect of collisions on latency using the added state approach .......... 144
7.31 Utilization using the added state approach ................................. 144
7.32 Effect of noise and collisions on the latency of a network using the added state approach .............................................. 145
7.33 The basic RSB simulink library block ........................................ 145
7.34 The Hardware block .............................................................. 147
7.35 The New A/D events block ...................................................... 147
7.36 The RX module block ............................................................ 148
7.37 The ACK management block .................................................. 148
7.38 How Simulink Performs Simulation .......................................... 149
7.39 The Retransmission block ....................................................... 152
7.40 FlowChart of ACK-control S-function ....................................... 153
7.41 The CSMA block ................................................................. 154
7.42 Communication channel used for 4 level 2 stations and 1 base station .... 154
7.43 Simulink network with 4 stations and one BS no multihop ............... 155
7.44 The Base station block .......................................................... 156
7.45 Settings dialog used to set the simulink simulation parameters ........ 157
7.46 RSB dialog used to identify the source and destination ID’s and to enable each station ......................................................... 157
7.47 Simulation results for a network with 4 level 2 stations and 1 BS with $\lambda_{station} = 1$, $t_{TTL} = 120ms$ and $t_{BO} = 120ms$ using the simulink model .............. 158
7.48 The effect that number of stations have on system latency vs $\lambda_{station}$ for the simulink model ................................................................. 159
7.49 The effect burst noise have on latency for the simulink model ................. 160
7.50 Effect multi-hop has on the simulink model where $\lambda_{station} = 0.5$, $t_{BO} = 20ms$ and $TTL_1 = TTL_2 = 60ms$ ......................................................... 161
7.51 Latency comparison between CSMA and conventional RRP .................. 162
7.52 Comparison between the CSMA model and the added state CSMA model, for a network with 10 stations and $t_{BO} = 20ms$ and $t_{TTL} = 60ms$ ................. 163
7.53 Comparison between the latency of the CSMA, conventional RRP and simulink model for a network with 4 level 2 stations and no additive noise ($t_{BO} = 20ms$ and $t_{TTL} = 60ms$) ............................................................... 164
7.54 Comparison between the latency for the CSMA and simulink model, for a network with 1 level 2, 4 level 3, 4 level 4 and 1 level 5 stations ($N_{mean} = 2.5$, $t_{BO} = 20ms$ and $t_{TTL} = 60ms$) ............................................................... 164
8.1 RF Inter-channel isolation ................................................................. 167
8.2 RF power settings ............................................................................ 168
8.3 Measured latency for a network with 4 level 2 stations .......................... 169
8.4 Measured data rate for a network with 4 level 2 stations ....................... 170
8.5 Measured latency for a network with 1 level 4 station .......................... 171
8.6 Measured data rate for a network with 1 level 4 station ........................ 172
8.7 Optimal service interval .................................................................... 173
8.8 Optimal mean back-off time ............................................................. 174
8.9 Measured effect of $\lambda_{station}$ on latency and percentage errors for a no-hop network 175
8.10 Measured effect of $\lambda_{station}$ on latency and percentage errors for a 3-hop network . 175
8.11 Comparison between the simulated and measured results for a no-hop network . 176
8.12 Comparison between the percentage packets lost between the simulink model and the measured results ............................................................... 176
8.13 Measured latency tests for 3 level 3 stations with 1 RSB only having one channel 177
8.14 Comparison between the prediction models and the measured values for a 2-hop network

8.15 Noise comparison between the simulink model and the measured results (2 level 3)

8.16 Comparison between the simulated and measured results for a 3-hop network

8.17 Noise comparison between the simulink model and the measured results (1 level 4)

A.1 Pin layout of the nRF905 transceiver

A.2 Flowchart ShockBurst\textsuperscript{TM} transmit of nRF905

A.3 Flowchart ShockBurst\textsuperscript{TM} receive of nRF905

A.4 Pin layout of the PIC24FJ128GA006 microcontroller

A.5 General block diagram of the PIC24FJ128GA microcontroller

A.6 Proposed connection circuit taken from the LM1117 datasheet

A.7 Proposed connection circuit taken from the MAX3232 datasheet
List of Tables

2.1 Complete list of the monitored points and their GPS coordinates . . . . . . . . 14
2.2 Proposed ground properties according to the Radio Mobile manual . . . . . . . 25
2.3 GPS coordinates of the added test points . . . . . . . . . . . . . . . . . . . . . 27

3.1 Routing codes for the network shown in fig. 3.5 . . . . . . . . . . . . . . . . . . 44
3.2 RF commandset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
3.3 Summary of the network . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51

4.1 Microcontrollers considered . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
4.2 Typical antenna properties taken from [Tec06] . . . . . . . . . . . . . . . . . . . 60
4.3 Total cost of one RSB PCB . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69

5.1 Memory map of the RTSP settings . . . . . . . . . . . . . . . . . . . . . . . . . . 83

6.1 Serial command set . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93

7.1 Table used for easy derivation of the state transition matrix . . . . . . . . . . . 143

8.1 Measured values . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 165
8.2 Important simulation variables . . . . . . . . . . . . . . . . . . . . . . . . . . . . 166

A.1 Electrical specifications taken from the nRF905 transceiver’s datasheet . . . . . 186
A.2 Operation modes of the nRF905 transceiver taken from the device datasheet . . 187
A.3 Pin functions of the nRF905 transceiver . . . . . . . . . . . . . . . . . . . . . . 188
A.4 DC characteristics of the PIC24FJ128GA006 taken from the device datasheet 192
# Nomenclature

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledge</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog to Digital Converter</td>
</tr>
<tr>
<td>ADO</td>
<td>ActiveX Data Objects</td>
</tr>
<tr>
<td>AM</td>
<td>Address Match</td>
</tr>
<tr>
<td>AP</td>
<td>Access Point</td>
</tr>
<tr>
<td>APB</td>
<td>Access Point Board</td>
</tr>
<tr>
<td>BO</td>
<td>Back-off</td>
</tr>
<tr>
<td>BS</td>
<td>Base Station</td>
</tr>
<tr>
<td>CD</td>
<td>Carrier Detect</td>
</tr>
<tr>
<td>CLK</td>
<td>Clock</td>
</tr>
<tr>
<td>CMD</td>
<td>Command</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CSMA</td>
<td>Carrier Sense Multiple Access</td>
</tr>
<tr>
<td>CSN</td>
<td>SPI Chip Select Not</td>
</tr>
<tr>
<td>DAQ</td>
<td>Data Acquisition</td>
</tr>
<tr>
<td>DEM</td>
<td>Digital Elevation Model</td>
</tr>
<tr>
<td>DIP</td>
<td>Dual-in-line Package</td>
</tr>
<tr>
<td>DR</td>
<td>Data Ready</td>
</tr>
<tr>
<td>EEPROM</td>
<td>Electrically Erasable Programmable Read-Only Memory</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>FCC</td>
<td>Federal Communications Commission</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correcting</td>
</tr>
<tr>
<td>GFSK</td>
<td>Gaussian Frequency Shift Keying</td>
</tr>
<tr>
<td>GP</td>
<td>Grand Parent</td>
</tr>
<tr>
<td>GPIO</td>
<td>General Purpose Input/Output</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System (also used to refer to a grandparent scan)</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>ICSP</td>
<td>In Circuit Serial Programming</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>ISM</td>
<td>Industrial-Scientific-Medical</td>
</tr>
<tr>
<td>ISR</td>
<td>Interrupt Service Routine</td>
</tr>
<tr>
<td>LDO</td>
<td>Low Dropout</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>LOS</td>
<td>Line-of-sight</td>
</tr>
<tr>
<td>LOS</td>
<td>Line of sight</td>
</tr>
<tr>
<td>MCU</td>
<td>Micro Controller Unit</td>
</tr>
<tr>
<td>MISO</td>
<td>SPI Master In Slave Out</td>
</tr>
<tr>
<td>MOSI</td>
<td>SPI Master Out Slave In</td>
</tr>
<tr>
<td>MSG</td>
<td>Message</td>
</tr>
<tr>
<td>NLOS</td>
<td>Non-line of sight</td>
</tr>
<tr>
<td>OLE DB</td>
<td>Object Linking and Embedding, Database</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase Lock Loop</td>
</tr>
<tr>
<td>PWR_DWN</td>
<td>Power Down</td>
</tr>
<tr>
<td>PWR_UP</td>
<td>Power Up</td>
</tr>
<tr>
<td>QFN</td>
<td>Quad Flat package No leads</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RRP</td>
<td>Round-Robin poling</td>
</tr>
<tr>
<td>RSB</td>
<td>Router Station Board</td>
</tr>
<tr>
<td>RTCC</td>
<td>Real-time Clock and Calendar</td>
</tr>
<tr>
<td>RTSP</td>
<td>Run Time Self Programming</td>
</tr>
<tr>
<td>RX</td>
<td>Receive</td>
</tr>
<tr>
<td>SCK</td>
<td>SPI Serial Clock</td>
</tr>
<tr>
<td>SOIC</td>
<td>Small-Outline Integrated Circuit</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>SPS</td>
<td>Samples per Second</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static random access memory</td>
</tr>
<tr>
<td>TQFP</td>
<td>Thin Quad Flat Pack</td>
</tr>
<tr>
<td>TRX_EN</td>
<td>Transmit/Receive Enable</td>
</tr>
<tr>
<td>TX</td>
<td>Transmit</td>
</tr>
<tr>
<td>TX_EN</td>
<td>Transmit Enable</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver Transmitter</td>
</tr>
<tr>
<td>WDT</td>
<td>Watchdog timer</td>
</tr>
<tr>
<td>WiMax</td>
<td>Worldwide Interoperability for Microwave Access</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Introduction and Background

Countless telemetry networks are already employed all over the world, monitoring and controlling various different physical parameters, from the sensors in racing cars to the water temperature of the oceans. Existing hardware for easy installation may be very expensive, as it is designed as a platform for a wide range of different applications. To be able to cover all the basic needs of different applications, the hardware often has a great deal of unnecessary functions and interfaces. The transmitters are often capable of transmitting in various frequency bands with transmit power of up to 1-5 W. The range of features and relatively high power transmitters are the main contributors to possible high cost. This high cost of telemetry hardware makes it impractical for some networks to be implemented.

An example of such a network was presented by an agricultural community approximately 30 km from Montagu in the Western Cape in South Africa. This community is supplied from a huge water network consisting of several bore-holes connected to each other via a pipeline. This pipeline is connected to various dams and water points which the farmers of the area can use as water supplies. The bore-holes connected to the pipeline were drilled by the Department of Water Affairs and are referred to as production bore-holes. The water levels and water flow rates in and out of the bore-holes and dams have to be monitored continuously, as must the pipeline pressure. Aside from monitoring the production bore-holes, the private bore-holes and fountains within the region must also be monitored to obtain the effect of water extraction on the surrounding environment. If extraction has a negative effect on a region, the pumps in the production bore-holes must be stopped. To be able to quantify the effect of extraction on the surrounding environment, the water out flow of the private bore-holes must also be measured to include the effect of local extraction. A map of the production bore-holes are shown in Fig. 1.1. Note that all the points to be monitored can be enclosed within a circle with a radius of approximately 10 km.
Most telemetry networks have one base station and a number of monitoring nodes or stations. All these stations have to transmit their data directly to the base station if they are within transmit range, or they have to transmit their data via predefined stationary repeater towers. This requirement, that each station must transmit directly to the base station, forces the use of high power transmitters to try and extend the maximum transmission range. The two furthest points in Fig. 1.1, are $\pm 20$ km apart. To implement a network in this area would require the use of stations which can transmit data at a distance of least 10 km. This area is also very rugged, featuring a couple of mountains and ridges, which makes it even more difficult to obtain good line-of-sight (LOS) links. To overcome the obstacles caused by the terrain, repeater towers are required. In total, there are more than 60 points that must be measured (see Table 2.1).

The main purpose of this thesis is to design a cheaper alternative for the above mentioned type of data acquisition. This includes the design of the hardware, software and the accompanying protocols. A feasibility study of the proposed hardware and software is also performed. The proposed protocol is analyzed and a software model, which enables us to predict the latency and throughput of this, and similar networks, is also presented. This software model is based on an existing model, but is significantly expanded to give more accurate results.

1.1.1 Reducing cost

Cost reduction is initiated by using an ad hoc network, rather than the traditional telemetry network explained above. The term ad hoc is a Latin phrase meaning “for this purpose”. The term ad hoc is, therefore, used to describe a solution that has been adapted or derived for a specific problem. A MANet (mobile ad-hoc network) is often used in mobile telecommunications to extend the range of the network by allowing each station to perform the tasks of both a monitoring station and a repeater. With a MANet, a station does not have to send its data directly to the BS, but it can send it via other nodes, as long as they are eventually connected to the BS. By using a MANet the minimum transmit power can be reduced enormously, which in turn will reduce the cost. The use of a MANet can also help to reduce the problem presented by the rugged terrain, as the stations would send the data to a station that would be able to communicate around the obstacle. If no way around can be found, the use of a repeater station is still required.

In general, telemetry networks are implemented using licensed frequencies to ensure interference free channels. The area under consideration is a rural area, with very little to no interference caused by other forms of wireless telecommunication. To further reduce cost, the license free 433 MHz, ISM band can be used for communication. The maximum transmit power allowed in this band is 100 mW, which is more than enough to ensure sufficient network coverage using a MANet.

Various low power RF controllers are available on the market, but they are usually used in remote controls for toys and other wireless applications. These chips are very attractive, as they tend to
be very cheap and easy to control and interface to. Relatively simple and cheap microcontrollers can be used together with these RF controllers to produce a low-priced, yet functional, solution. Before such a station can be designed, a feasibility study has to be undertaken to determine whether such a station would provide adequate radio coverage and data handling abilities.

1.1.2 Feasibility study

The proposed RF controller has a maximum transmit power of 10 dBm. To allow the use of this controller, it must be shown that 10 dBm transmit power at 433 MHz would be sufficient to cover the desired network shown in Fig. 1.1. This study is described in Chapter 2.

1.1.3 Prediction model

To be able to predict the performance of a protocol to some certainty, a mathematical model of that protocol must be derived. Different protocol options must be weighed against each other to try and determine which protocol is best suited for this type of data acquisition network. As explained in Chapter 7, the protocol chosen for this network is based on a CSMA strategy. The current mathematical model for CSMA is based on numerous assumptions, especially when it involves noise and other sources of errors. In this thesis, modeling of the actual sources of errors is attempted, to try and obtain a more quantified approach to error handling in CSMA type networks. The effect of retransmissions on the utilization of the network is also not taken into account with the currently available models. In this thesis, this aspect will also be studied, as it is very important in determining network stability and congestion.

To the best of this author’s knowledge, the above-mentioned expansions to the normal CSMA model have not yet been studied or formalized in any previous publications. The complete derivation of this model is discussed in Chapter 7.

1.1.4 Summary and Contributions

In this thesis:

- The feasibility of the practical application of an ad-hoc protocol to reduce cost and to simplify network planning, is examined;
- The optimal protocol for the desired network is determined;
- A successful performance prediction algorithm is derived by expanding a current popular theoretical model;
- The proposed solution is implemented in hardware;
• Results obtained by measurement are also presented, in confirmation of the developed theory and as proof of the feasibility of the proposed solution.

1.2 Overview of the thesis

The rest of the thesis, is set out as follow:

• Chapter 2 - Network Design

Before any hardware or software can be designed, it should be determined whether the proposed RF transceiver would provide sufficient network coverage or not. In this chapter the basic principles of free space radio propagation are discussed providing the reader with adequate background, in the understanding of the functioning of RF communication links. The link budgets of all the nodes are also determined using radio propagation software. The link budget results are discussed, together with the adjustments and additions needed to obtain a viable solution. This chapter will show that the proposed transceivers would supply sufficient RF power.

• Chapter 3 - Communication Strategy

In this chapter the complete communication strategy for this network is discussed, based on the 7 layer OSI model. To be able to design this strategy some information from other chapters is required. This information includes the optimal basic data link protocol for this network, which Chapter 7 shows to be CSMA. It also includes the architecture of the hardware, which is discussed in Chapter 4. This chapter also describes how new stations are added to the network and how data is routed throughout the network.

• Chapter 4 - Hardware Design

This chapter describes the design process of the hardware, including motivations for using each component. The circuitry required to convert the transducer measured signal to an analogue voltage interpretable by the microprocessor, was not added to the hardware board. The reason for this is based on the fact that the output for different transducers vary individually and they should be added externally according to the transducer used. The typical transducer circuitry required is explained. A complete cost analysis of the hardware is also derived in this chapter and is shown in Table 4.3.

• Chapter 5 - Embedded PIC software

The designed hardware requires embedded controlling software. The design of this code is explained in this chapter. All the interfaces of the hardware were programmed to be interrupt driven. This allows the hardware to pause sometimes, while waiting for a new task. In doing this, the average time a task would wait before being serviced is reduced, as the microcontroller does not have to poll all peripherals to check if they have a task or not.
• Chapter 6 - Server Software (GUI)

The server software are responsible for all data logging and network control. To simplify the interface with the server, a graphical user interface (GUI) was designed to enable the user to control the network using a visual interface. The server GUI also contains an additional interface which can be used to calculate the network performance. The design of this GUI software is explained in this chapter.

• Chapter 7 - Performance Predictions

This chapter starts with a brief background study of queueing theory, which is required in later analysis. Round-robin polling (RRP), which is a member of the collision free protocols family, is then analyzed mathematically using both normal analytical math and queueing theory. The results obtained for both these analyses are then discussed and compared. After this, the CSMA contention protocol is also analyzed using queueing theory. The same approach followed by [Wol02] and [Nic06] is used, but the model is expanded further to incorporate burst noise and collisions caused by both hidden terminals and transceiver rise times. The standard approach used to compensate for collisions was replaced with a statistical analysis of these phenomena. A simulink model of the CSMA protocol was also designed and compared to that of the mathematical analysis. The results obtained from the analysis of the RRP and CSMA protocols were compared to determine the most suitable protocol.

• Chapter 8 - Measurements and Results

The network performance measured with the GUI, described in Chapter 6, is compared to the predicted values obtained from both the mathematical model and the simulink model, as described in Chapter 7. The efficiency of the network is also determined in this chapter.

• Chapter 9 - Summary and Conclusions

A summary of the complete thesis is given in this chapter, together with some concluding remarks. Recommendations for future work are also mentioned here.

• List of References

• Appendices

Some additional information regarding the design process of the hardware and software is given in the appendices. The additional information is classified according to:

− Hardware design information

  Appendix A contains excerpts taken from data sheets which were essential to the design of the hardware.

− DVD-ROM guide

  Appendix B contains a hierarchy of all the files saved on the accompanying DVD. This includes datasheets, program code, software and digital elevation maps.
Figure 1.1: Supplied map of the area
Chapter 2

Network Design

In this chapter the link properties of the desired network will be discussed. The design of a MATLAB program which can be used to determine the performance of these links will be explained. Other design tools used to analyze the network layout will also be mentioned. Before these links can be analyzed, some background information is required.

2.1 Basic propagation principles

To better understand RF communication in this case, the basic principles of communication have to be revisited. The first principle to be described is that of free space propagation.

2.1.1 Free space propagation

The free space path loss in a transmission link is a measure of how much the link quality would decrease in a region which is free of any obstacles that might absorb or reflect radio energy (free space). The free space loss can be determined using eq. 2.1, which was derived in [McL01]. The Friis equation is used to determine the received power. The path loss equation can then be obtained by calculating $P_t / P_r$. In this equation $c$ is the speed of light ($m/s$), $f$ is the frequency (Hz) and $d$ is the distance between the transmitter and receiver (m). Free space path loss is usually expressed in logarithmic form as shown in eq. 2.2. In this equation $f$ is the frequency in MHz and $d$ is the distance in km.

$$L_p = \left(\frac{4\pi}{c}\right)^2 f^2 d^2$$  \hspace{1cm} (2.1)

$$L_p = 20 \log_{10}(d) + 20 \log_{10}(f) + 32.44$$  \hspace{1cm} (2.2)

The first equation clearly shows that the transmission loss is proportional to the square of the distance. In logarithmic terms it can be stated that the loss will be increased by 6 dB every time the distance between the transmitter and receiver is doubled. It should also be noted that the
equations given thus far, only consider the path loss and do not take the actual antenna gains and cable losses into account. The actual power received at the receiver is given by eq. 2.3.

\[
P_r = P_t - L_p + G_t + G_r - L_t - L_p
\]  

(2.3)

Where:

- \( P_t \): transmitted power (dBm)
- \( L_p \): free space path loss between isotropic antennas (dB)
- \( G_t \): transmit antenna gain
- \( G_r \): receive antenna gain
- \( L_t \): transmission line loss between transmitter and transmit antenna (dB)
- \( L_r \): transmission line loss between receive antenna and receiver (dB)

Free space propagation rarely exists. To further analyze transmission links, the different links are divided into two types: line of sight links (LOS) and non-line of sight links (NLOS).

### 2.1.2 Line of sight (LOS) links

Line of sight (LOS) links refers to those links where the transmitting antenna has an unobstructed link with the receiving antenna. Note that the radio horizon is not exactly the same as the optical horizon, as radio signals tend to follow slightly curved paths around the earth’s natural curvature. These curved paths are caused by refraction. Refraction is not that prominent with short distance links where the earth’s curvature can be approximated as a flat surface. If an antenna has an unobstructed link with another antenna via this curved path, it is still referred to as a LOS link. Even though it may seem as if the path loss of a LOS link should be the same as that of a free space link, this is not always true. Refraction, diffraction and reflection of signals can cause the path loss to deviate from that of a normal free space link. The cause and effect of these three processes will now be described. Note that these are not the only sources of additional loss, but are the major components that should be taken into account in this design.

#### 2.1.2.1 Atmospheric refraction

Atmospheric refraction refers to the change in direction of a wave due to a variation in air density, which causes the wave’s propagation speed to change. The index of refraction decreases with an increase in height, which causes radio waves to bend slightly towards the earth’s surface rather than follow a straight line. Radio frequencies are influenced much more than those of visible light, which allow radio signals to propagate beyond the optical horizon without any additional losses other than free space loss. The index of refraction can change with a change
in temperature, weather conditions and any other natural changes. The transmission path of a signal is, therefore, not always constant and can change as a function of the weather. Superrefraction is a term used to describe the case where the radio wave bends more than is normal, causing the radio horizon to extend even further than normal. With subrefraction the radio horizon is shortened, causing the clearance of obstacles to be reduced. The latter case can cause major problems. Refraction renders link design difficult, as it cannot be modeled by straight lines. To compensate for this phenomenon an approximate method, often referred to as the 4/3 earth radius approximation, is often used. With this approximate method, the earth’s radius is increased by 4/3, which allows the use of straight lines to represent transmission links.

Safety margins, or fading margins, should be implemented when designing a communication link, to compensate for the different refraction scenarios which might occur. The network under consideration does not require the use of long data links, which reduces the effect of refraction.

### 2.1.2.2 Diffraction and Fresnel zones

The principle of diffraction was discovered by a Dutch physicist, Christian Huygens, in 1678. This principle is called Huygens’ principle and describes how waves are able to bend around obstacles (see [SB00]). According to [SB00], Huygens’ principle can be stated as: All points on a given wave front are taken as point sources for the production of spherical secondary waves, called wavelets, which propagate outward through a medium with speed characteristics of waves in that medium. After some time has elapsed, the new position of the wave front is the surface tangential to the wavelets. Diffraction basically describes what happens to a wave when it encounters an obstacle. To be able to visualize the wave after colliding with an obstruction, we have to develop a picture of the wavefront as a function of the wave before it collided with the obstacle. This can be very troublesome, but is assisted enormously by the introduction of Fresnel zones.

The Fresnel Zone is named after a French physicist, Augustin-Jean Fresnel, who described this principle in 1818. The Fresnel Zone is the volume of space enclosed by an ellipsoid, where the transmit and receive antennas are placed at the ends of the radio link at its foci. This Fresnel Zone is shown in Fig. 2.1. The surface of the ellipsoid is defined by the path ACB, which exceeds the length of the direct path AB by $\lambda n / 2\pi$, where $n$ is a positive integer (see [McL01] and [RW06]). The first Fresnel zone is obtained when $n = 1$, which results in path ACB being $\lambda / (180^\circ)$ longer than AB. The use of Fresnel Zones makes it easy to maximize the received signal strength, by minimizing the effect of out of phase signals caused by obstacles in the RF LOS, because the strongest signals are on the direct line between the antennas and always lie within the first Fresnel Zone.

When designing radio links at least 60% of the first Fresnel zone must be clear, to obtain a resultant close to that of free space propagation. To determine the loss caused by an obstruction, we use Fig. 2.2. Note that this figure only represents a two dimensional solution, but the Fresnel
Zone is completely three dimensional. The same approach can be followed to calculate horizontal obstacles. Only LOS links are inspected in this section, which limits links to only those having negative obstacles (Figure 2.2 b). The minimum clearance \( (h) \) needed between the direct link and the obstacle can be derived to be approximated as that of eq. 2.4. In this equation \( d_1 \) and \( d_2 \) are the distances between the obstacle and the transmit and receive antennas respectively, and \( f \) is the frequency in GHz.

\[
h = 17.3 \sqrt{\frac{d_1 d_2}{f (d_1 + d_2)}}
\]  

(2.4)
Note that the derivation of Fresnel losses described thus far is based on a knife edge scenario, where the top of the obstacle is small in terms of wavelength. This is seldom the case, but this derivation gives a good idea of the effect obstacles have on communication links. The approximate diffraction loss \( L_{fz} \) caused by Fresnel intrusion can be determined using eq. 2.6, where the diffraction parameter \( v \) is given by eq. 2.5. The diffraction parameter is positive when the direct path is blocked, and negative when the direct path is not obstructed.

\[
v = 2\sqrt{\frac{\Delta d}{\lambda}}, \quad \text{with} \quad \Delta d = d_1 + d_2 - d \quad (2.5)
\]

\[
L_{fz} = 6.9 + 20\log_{10} \left[ \sqrt{v^2 + 1} + v \right] \quad (2.6)
\]

In real life problems the topography of the obstacles tends to be more complex than a knife edge. Numerous different approximate methods have been developed to try and model these different topographies. A commonly used model is that of the single radius method, where the top of the intrusion is modeled as a cylinder with radius \( r \). A graphical explanation of this model is shown in Fig. 2.3. To calculate the added loss due to this intrusion, the profile of the object must be plotted, with straight lines drawn from the end points to the top of the intrusion, so that it just clears the object. The estimated radius \( r \) can now be determined using eq. 2.7, which relies on the use of the estimated distances \( D_s, d_1 \) and \( d_2 \). Note that angle \( \alpha \) is measured in radians.

\[
r = \frac{sD_sd_1d_2}{\alpha (d_1^2 + d_2^2)} \quad (2.7)
\]

**Figure 2.3:** Diffraction of a rounded obstacle (image taken from [McL01])
is that smoother surfaces tend to produce higher diffraction losses. If a hill is covered in trees or is very rugged, this would result in a lower diffraction loss. To compensate for the terrain a roughness factor can be added. According to [McL01], a hill covered with trees would typically produce only approximately 65% of the diffraction loss when compared to a smooth hill.

\[ L_{ex} = 11.7a \sqrt{\frac{\pi r}{\lambda}} \]  

(2.8)

### 2.1.2.3 Ground reflections

Even with no obstacles in the Fresnel Zone, the path loss still differs from that of free space due to multipaths. The multipath phenomenon is caused by reflections from surfaces. The ground is a major source of reflection, especially in rural areas where few or no obstacles such as buildings or trees are present. When observing a terrain to determine sources of reflection, it should be remembered that the angle of incidence equals the angle of reflection. The influence of reflections can be either positive or negative, depending on the amplitude and phase of the reflected signal.

### 2.1.3 Non-line of sight (NLOS) links

NLOS links are not ideal, but it is impossible to avoid them in some cases. With NLOS links no clear radio path is available for transmission, but the path is partially obstructed, usually by a physical object in the Fresnel Zone. Typical obstacles include buildings, trees, mountains or high voltage electric power lines. Obstacles can either reflect certain frequencies or they can absorb or distort a signal. To overcome the problem of NLOS the signal can either be relayed around the obstacle, or multipath can be used to obtain a link. In some cases the RF signal would be strong enough to penetrate through the obstacle to reach the receiver. This typically happens when a signal has to be propagated through trees, or even forests. It should be stated that the attenuation caused by trees depends on numerous variables such as dampness, the type of tree and whether they have leaves or not. When propagating signals through forests, the typical attenuation that can be expected is roughly 0.05 dB/m at 200 MHz, 0.1 dB/m at 500 MHz and 0.2 dB/m at 1 GHz.

With NLOS links ground reflections are usually not an issue, but refraction and diffraction still occur. The diffraction can be determined using the same method described for the LOS case, but it should be remembered that a positive obstacle (scenario a in fig. 2.2) must be used when analyzing the knife edge Fresnel diffraetion. The total loss for the NLOS case is very difficult to analyze and, therefore, the use of these links should be avoided as far as possible.
2.1.4 Software used for propagation predictions

As explained thus far, it is very difficult to predict the link quality using a mathematical approach taking terrain characteristics fully into account. Various software programs have been developed which can be used for propagation predictions. RFProp is a very basic program developed by Colin Seymour, which determines the effect of obstacles on a propagated signal based on the single radius method described in Section 2.1.2.2. Even though this program is not that advanced, it gives a fairly good idea of the effect of diffraction on a communication link in the presence of an obstacle. With this program an actual profile cannot be used.

More advanced programs exist which can acquire terrain information from digital elevation model (DEM) files. A digital elevation model (DEM) is a digital representation of ground surface topography or terrain. A DEM can be stored in many different formats. Commonly used DEM formats are: USGS (United States Geological Survey), SDTS (Spatial Data Transfer Standard), DTED (Digital Terrain Elevation Data), SRTM (Shuttle Radar Topography Mission) and GTOPO30. When using DEM files the actual topography for the network can be used when determining radio propagation. These programs are usually quite expensive, but some free versions are also available. Software that can be purchased includes Pathloss, RadioLink, PathCalcUK, Siradel and Volcano. Radio mobile is a freeware program designed by Roger Coudé which also uses DEM files to acquire the necessary topography. Radio mobile can be downloaded from http://www.cplus.org/rmw/english1.html.

2.2 Network design

In the previous section the basic design of a communication link was discussed. A network consists of a number of stations, requiring multiple links. To design a network each link must be designed individually. The end result requires that each station must be connected directly or indirectly to each of the other stations. A major problem with this specific project, is the rugged terrain of the area, which requires the use of multihop to access all the different stations. This can be achieved by using either fixed repeater stations or some sort of ad hoc network, where stations can route through other stations to reach their destination. The latter solution is proposed, but it should first be verified that such a solution would be adequate. The main motivation for using an ad hoc network is the easy deployment of such networks, as well as its ability to be expanded without requiring any changes or adjustments to current networks. A new station can simply be powered up anywhere, from where it would find its own path to the network, provided that it is within transmission radius of any station that is connected to the network. Another important design specification is that of cost reduction.

To be able to analyze this network, the GPS coordinates of the points to be measured are required, as well as some DEM of the terrain. A list of all the points and their corresponding GPS coordinates is given in Table 2.1. The SRTM format, which can be downloaded freely from
[NAS06], was used to create the DEM of the area. SRTM is an international project spearheaded by the U.S. National Geospatial-Intelligence Agency (NGA) and the U.S. National Aeronautics and Space Administration (NASA) ([NAS06]). The SRTM format is the most complete high-resolution digital topographic database of Earth to date, covering the globe from 56°S to 60°N. The resolution of the data is three arc seconds, which is equivalent to approximately 90 m. The SRTM data is divided into tiles which cover 1 degree longitude and 1 degree latitude. The dimensions of a 3 arc seconds tile are 1201 × 1201.

<table>
<thead>
<tr>
<th>Name</th>
<th>GPS coordinates S</th>
<th>E</th>
<th>Name</th>
<th>GPS coordinates S</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>02/13</td>
<td>33°36′39″</td>
<td>19°43′38″</td>
<td>EN9</td>
<td>33°38′04″</td>
<td>19°52′07″</td>
</tr>
<tr>
<td>02/24</td>
<td>33°36′14″</td>
<td>19°43′44″</td>
<td>EN10</td>
<td>33°37′57″</td>
<td>19°52′06″</td>
</tr>
<tr>
<td>02/23</td>
<td>33°36′14″</td>
<td>19°43′45″</td>
<td>EN11</td>
<td>33°37′50″</td>
<td>19°52′15″</td>
</tr>
<tr>
<td>02/01</td>
<td>33°37′47″</td>
<td>19°45′39″</td>
<td>EN12</td>
<td>33°37′43″</td>
<td>19°52′13″</td>
</tr>
<tr>
<td>02/05</td>
<td>33°37′48″</td>
<td>19°45′42″</td>
<td>EN13</td>
<td>33°37′54″</td>
<td>19°52′38″</td>
</tr>
<tr>
<td>02/02</td>
<td>33°38′14″</td>
<td>19°45′05″</td>
<td>EN15</td>
<td>33°37′53″</td>
<td>19°52′44″</td>
</tr>
<tr>
<td>02/06</td>
<td>33°38′55″</td>
<td>19°48′00″</td>
<td>EN16</td>
<td>33°38′07″</td>
<td>19°52′03″</td>
</tr>
<tr>
<td>02/11</td>
<td>33°35′30″</td>
<td>19°47′37″</td>
<td>EN17</td>
<td>33°37′06″</td>
<td>19°43′36″</td>
</tr>
<tr>
<td>02/15</td>
<td>33°36′36″</td>
<td>19°49′11″</td>
<td>EN18</td>
<td>33°38′56″</td>
<td>19°47′58″</td>
</tr>
<tr>
<td>02/04</td>
<td>33°37′05″</td>
<td>19°49′50″</td>
<td>EN19</td>
<td>33°37′10″</td>
<td>19°49′49″</td>
</tr>
<tr>
<td>02/08</td>
<td>33°37′35″</td>
<td>19°50′25″</td>
<td>EN20</td>
<td>33°34′53″</td>
<td>19°51′02″</td>
</tr>
<tr>
<td>02/21</td>
<td>33°37′47″</td>
<td>19°53′04″</td>
<td>EN21</td>
<td>33°37′43″</td>
<td>19°52′20″</td>
</tr>
<tr>
<td>02/16</td>
<td>33°37′58″</td>
<td>19°52′34″</td>
<td>EN22</td>
<td>33°37′51″</td>
<td>19°45′38″</td>
</tr>
<tr>
<td>HG6</td>
<td>33°38′05″</td>
<td>19°45′27″</td>
<td>EN23</td>
<td>33°38′16″</td>
<td>19°45′26″</td>
</tr>
<tr>
<td>BL1</td>
<td>33°38′17″</td>
<td>19°45′30″</td>
<td>EN24</td>
<td>33°40′41″</td>
<td>19°48′58″</td>
</tr>
<tr>
<td>02/20</td>
<td>33°38′50″</td>
<td>19°46′50″</td>
<td>EN25</td>
<td>33°38′51″</td>
<td>19°46′46″</td>
</tr>
<tr>
<td>LR4</td>
<td>33°38′45″</td>
<td>19°47′23″</td>
<td>EN26</td>
<td>33°47′41″</td>
<td>19°47′41″</td>
</tr>
<tr>
<td>LTR3</td>
<td>33°39′25″</td>
<td>19°47′35″</td>
<td>EN27</td>
<td>33°39′42″</td>
<td>19°47′47″</td>
</tr>
<tr>
<td>LM7</td>
<td>33°39′22″</td>
<td>19°48′22″</td>
<td>EN28</td>
<td>33°40′03″</td>
<td>19°48′16″</td>
</tr>
<tr>
<td>LM4</td>
<td>33°39′44″</td>
<td>19°48′23″</td>
<td>EN29</td>
<td>33°40′44″</td>
<td>19°49′04″</td>
</tr>
<tr>
<td>LM1</td>
<td>33°40′00″</td>
<td>19°48′14″</td>
<td>EN30</td>
<td>33°41′51″</td>
<td>19°51′15″</td>
</tr>
<tr>
<td>LM3</td>
<td>33°40′52″</td>
<td>19°48′21″</td>
<td>EN31</td>
<td>33°41′51″</td>
<td>19°51′15″</td>
</tr>
<tr>
<td>02/07</td>
<td>33°40′44″</td>
<td>19°48′50″</td>
<td>EN32</td>
<td>33°41′51″</td>
<td>19°51′12″</td>
</tr>
<tr>
<td>JI7</td>
<td>33°40′39″</td>
<td>19°49′22″</td>
<td>EN33</td>
<td>33°41′44″</td>
<td>19°50′41″</td>
</tr>
<tr>
<td>02/18</td>
<td>33°35′38″</td>
<td>19°47′57″</td>
<td>EN34</td>
<td>33°40′59″</td>
<td>19°49′35″</td>
</tr>
<tr>
<td>02/17</td>
<td>33°35′46″</td>
<td>19°48′27″</td>
<td>JI7</td>
<td>33°40′48″</td>
<td>19°49′21″</td>
</tr>
<tr>
<td>RE2</td>
<td>33°37′48″</td>
<td>19°49′16″</td>
<td>JI6</td>
<td>33°40′00″</td>
<td>19°49′35″</td>
</tr>
<tr>
<td>BL2</td>
<td>33°37′28″</td>
<td>19°47′55″</td>
<td>JI7</td>
<td>33°40′19″</td>
<td>19°49′44″</td>
</tr>
<tr>
<td>PA2</td>
<td>33°40′37″</td>
<td>19°50′33″</td>
<td>VERSA1</td>
<td>33°41′22″</td>
<td>19°50′00″</td>
</tr>
<tr>
<td>TS3</td>
<td>33°41′06″</td>
<td>19°50′48″</td>
<td>VERSA2</td>
<td>33°41′33″</td>
<td>19°50′13″</td>
</tr>
<tr>
<td>LHK1</td>
<td>33°39′33″</td>
<td>19°50′24″</td>
<td>EIKAF1</td>
<td>33°41′33″</td>
<td>19°50′13″</td>
</tr>
</tbody>
</table>

Table 2.1: Complete list of the monitored points and their GPS coordinates

The proposed hardware consists of a transceiver with a maximum transmission power of 10 dBm and a receiver sensitivity of -100 dBm, operating in the 433 MHz ISM band. According
to [Tec06] and [SKHH06] this type of transceiver would typically be able to transmit up to 1-2 km over a LOS link. As mentioned in Section 2.1, it is very difficult to predict the propagation losses of a NLOS link. The most reliable network would be a network that consisted only of LOS links. It is not that difficult to find an optimal solution manually for one link using the DEM information, but it can become quite time consuming and complex when a complete network has to be designed. It should be taken into account that the topography is completely 3 dimensional and that an adjustment to 1 antenna to create an acceptable link, can cause another link to break. The proposed idea is, therefore, to try and create a program which can automatically find the optimum position for each antenna, based on LOS links.

### 2.2.1 Software based design of optimal radio links

The mapping toolbox used by MATLAB makes it very easy to manipulate and plot GPS data and other positioning formats. It also has numerous navigation functions, which is especially handy when the GPS coordinates between two points are required, or if a GPS coordinate is required for a point which is a certain distance in a given direction from another point. Sebastian Hölz created an m-file (*GetSRTMDdata*) which can be used to extract elevation data from 3 arc second SRTM files, and which can be downloaded from the MATLAB file exchange sever ([fe]). Note that the GPS coordinates input to this function must be specified as decimal degree coordinates (DEG), while the notation used in Table 2.1 is in degrees-minutes-seconds (DMS) format. The DMS data can be converted to the DEG format using the *dms2deg* function. It is also important that the relevant SRTM files (.hgt) should be available in the MATLAB path directory. When using the *GetSRTMDdata* function, the user does not have to know which SRTM file is needed, because this function automatically determines which file should be used for data extraction.

MATLAB also has various functions which allow easy migration of data between its workspace and Microsoft Office Excel worksheets. This allows the use of a very simple graphical interface for the input and output of data. An Excel worksheet (named *data_points.xls*) is now created which contains all the necessary information and settings of each station. This worksheet must also reside within the MATLAB path directory. A list of all the columns used in this worksheet are given below. Some of the settings used have not been mentioned, but will be explained later in the chapter.

- **HANDLE** - This section consists of two columns which are used as handles to refer to the different stations.
  - **NO** - Used as a short notation to refer to the different station
  - **Name** - The unique name of each of the stations

- **DATA POINT INFO** - This section consists of three columns and is used to describe the position of each of the stations.
– **Status** - If this field is 0, it indicates that the station should not be used, while a 1 indicates that the station is in working order. The use of this status flag enables the user to test the response of the network if some of the stations should fail.

– **Lat** - GPS latitude of the point under consideration in DMS format.

– **Lon** - GPS longitude of the point under consideration in DMS format.

• **ROUTING INFO**

– **Level** - The level of each station based on the routing protocol discussed in Section 3.3.1. Note that at least 1 station must be set as the BS (level 1) to be able to initialize a network setup.

– **Range (km)** - The maximum transmission range in the case of a LOS link.

– **Ant_Height (m)** - Height of the stations antenna.

• **RADIO MOBILE SETTINGS** - These 4 columns describe the marker properties used when the data points are imported to Radio Mobile. See Section 2.2.1.2 for more details.

– **icon** - The index in the icon list corresponding to the preferred icon to use.

– **Forecolor** - The foreground colour of the icon.

– **Transparent** - Transparency of the icon.

– **Backcolor** - The background colour of the icon.

• **ANTENNA PROPERTIES**

– **Frequency [MHz]** - The frequency used for communication. This field allows the user to specify the frequency band used, to enable the use of different networks in the same area. Note that the MATLAB program will only connect stations that are transmitting at the same frequency.

– **TX-power [dB]** - Transmitted power.

– **RX-sensitivity [dBm]** - Transceiver sensitivity.

All the data points with their settings must be added to the worksheet described above before the network link analysis can begin. The basic flowchart of the MATLAB program is shown in Fig. 2.4. The first block is very important and describes the maximum allowable adjustment settings. These settings are:

• **hight_max** - maximum allowed height of an antenna

• **reduce_ant_high** - increments in which antenna masts must be lengthened if required

• **gridsize** - step size in the profile between two points when testing LOS

• **dist_tol** - the maximum distance allowed to move a point to gain LOS
The next step in this flowchart is the importing of all the data and settings from the Excel worksheet. With this program the user can choose whether he wants the program to plot all the profile schematics for the LOS links (input argument `plot_connected_links`), and whether he wants the program to auto adjust antennas to try and gain LOS links (input argument `autoset`). After analyzing all the links, the final network results are plotted before exporting the results to a Radio Mobile compatible file (`RadioMobile.txt`) and a Kashmir 3D file (`Kashmir.wpt`). Kashmir 3D is a very handy visual tool which can be used to view the DEM in a 3 dimensional environment. Even though this program cannot perform any propagation calculations, it can profitably be used to obtain some insight of the terrain. This is also a freeware program which can be downloaded from http://www.kashmir3d.com/index-e.html. The process used to analyze links will now be discussed in more detail.

2.2.1.1 Analyzing links

In this section the two “determine links” blocks shown in Fig. 2.4, will be further refined. To be able to analyze all the links the MATLAB program requires the correct SRTM files and the data contained in the `data_points.xls` worksheet. Note that at least one of the stations must be selected to be a BS (level 1). The link analyzing procedure starts by searching through the list of

- step_size - step size when looking for higher ground in a certain direction
all the stations for the level 1 stations. These stations are then used to build the network. The linking process works exactly the same as the routing process explained in Section 3.3.1. Each station will have only 1 parent station, but can have multiple children. It should be remembered that this program is only intended to determine if the proposed solution is viable and, therefore, we are only interested in trying to link as many stations as possible. We are not interested in finding the best link, but the maximum number of linked stations. The linking can now be summarized as that all level \( n \) stations can be linked to only one level \((n - 1)\) station and to multiple level \((n + 1)\) stations, while level 1 stations can also be linked to each other, as they are the backbone of the network.

To be able to link two stations, they must have a LOS link which is within the maximum transmission range specified for both stations (Range (km) in the data_points.xls worksheet). This program would start with the first level 1 stations and would then connect all the stations that satisfy the previous statement, to itself. All the other level 1 stations would then do exactly the same, to try and connect the rest of the unconnected links to themselves. When all the level 1 stations have finished connecting, the level 2 stations would be allowed to try and establish further links with level 0 stations. This procedure would traverse through out all the linked stations until no new stations can be linked. A flowchart of the linking procedure is shown in Fig. 2.5.

This linking process is exactly the same for all stations. In this flowchart station A is a station that is already linked to the network (level > 0), while station B is an unrouted station that is currently not linked (level = 0). If station A is a BS, then this procedure would occur if B is either a level 0 or level 1 station. The first test is to determine if the two stations are within transmission range of each other. Note that the transmission range \( d \), coincides with the transmission range of the station with the shortest range. If the two stations are within range of each other, this program would determine whether a LOS link exists. This LOS test is performed by drawing a straight line between the two points and then extracting the elevation data of numerous points situated on the line. The gridsize (units in km) variable defines the interval of extraction. With this extracted data, a profile schematic of the link can be constructed. An example of such a profile is shown in Fig. 2.6. Note that this profile shown is that of the link between two WiMax stations, which had been considered for use as a backbone during the initial design phase. For higher accuracy the value of the gridsize variable should be reduced. Note that the Fresnel Zone is also computed using eq. 2.4 and is used to determine whether the desired clearance is obtained or not. As mentioned earlier only 60% of the first Fresnel Zone (also denoted 0.6\( F_1 \)) has to be cleared to produce a link close to that of free space.

If a LOS link exists the link is added to the link list, before the analysis of the next link starts. If a LOS link does not exist the program would either try to find a solution, if the autoset function is enabled, or it would proceed to the next link. During the rest of the analysis it is assumed that the autoset function is enabled. This program is able to adjust two parameters to try and obtain a LOS link. It can either change the antenna height of stations A and B, up
Figure 2.5: Flowchart of the link analysis procedure between two stations
to a maximum of \( \text{height}_\text{max} \) (units in [m]), or it can change the GPS position of the point by moving the antenna with a maximum of \( \text{dist}_\text{tol} \) [m] from the original position, or a combination of both can be used. The first scenario is the easiest solution and is, therefore, tested first. Both station’s antennas are increased to the maximum allowable height and then tested for a LOS link. If a LOS link is obtained by this change, the program would slowly decrease the height of the antennas until a LOS link with the shortest possible antenna masts is obtained. If this fails, the program would try moving station B’s antenna around within a radius of \( \text{dist}_\text{tol} \) from the current position. This is done by taking the GPS position of the point that is exactly \( \text{dist}_\text{tol} \) meter north (N) of the current position. A line is now drawn between these two points and the highest point obtained. The antenna is now moved to this point the link is tested for LOS. If no LOS is obtained the antenna lengths are adjusted to try and obtain a link. If a link is obtained the distance moved and the antenna adjustments are stored in memory. The same procedure is now repeated but the antenna is moved in all the other major directions [NW, W, SW, S, SE, E and NE]. After moving the antenna to the highest position in all 8 directions, the program would choose the best solution, requiring the least adjustment.

After all the links have been analyzed, a list is generated of all the stations that are not connected. If the \textit{autoset} option is chosen, the program would also link all the remaining level 0 stations that have LOS links with each other. This creates clusters that contains only level 0 stations. The program would then find the closest route for each level 0 station to another station that
is connected to the network. The last link task to perform is to look through all the shortest links in a cluster that connects the level 0 stations to the network, and then determine the best available link to try and connect that cluster to the network. It should be noted that all the different links are stored in a variable, even if they are not viable, but only the profiles of LOS links are stored. A link type is assigned to each link in order to describe it. In total seven different link types exist and are denoted type 1 through to type 6 and then types greater than 10. The different types are:

1. A LOS link between a BS and another BS. These links are displayed in the final network plot as thick dashed red lines.

2. A LOS link between a BS and a level 2 station. These links are displayed in the final network plot as thick dashed blue lines.

3. A LOS link between any two stations where the level of both stations is greater than 1. These links are displayed in the final network plot as thin dashed green lines.

4. A special LOS link between two level 0 stations. These links are displayed in the final network plot as thin dotted magenta lines.

5. A special out of range link between a level 0 station and another station connected to the network. These links are displayed in the final network plot as thick dotted black lines.

6. Represents the best link to try and implement to connect a cluster. These links are displayed in the final network plot as thin dotted magenta lines.

7. This type of link represents NLOS links, but is not denoted type 7, but is represented by values greater than 10. The value is derived by adding 10 to the type of link described above (11-16). These links are displayed in the final network plot as thin dotted magenta lines.

During execution of the code, the program displays the current states of the links in four different Java boxes. These boxes are displayed using the \texttt{jprintf} function, which was designed by Mark Brown, and can be downloaded from [fe]. The first box contains a list of all the possible antenna auto corrections which would result in LOS links, while the second box displays all the LOS links that have been obtained thus far. The third box displays a list of the proposed best solution adjustments required to obtain LOS links. The last box displays all the unconnected stations.

At the end of execution, this function returns a structure that contains further station structures for each station present in the network. Each station structure contains all the columns specified in the previously discussed worksheet (\texttt{data_points.xls}), as well as an elevation field and an additional \texttt{Links} structure. The \texttt{Links} structure contains a list of all stations that it is linked to (the \texttt{NO HANDLE} is used). The level of each of these linked stations is also stored, together with a type variable classifying each link, and a field containing the distance of each link. In the
case of LOS links the profile of the link is also stored. This variable allows the user to further analyze all the links if needed (post processing). MATLAB also allows the user to save variables (using the .mat extension) and to import previously stored variables. This functionality allows the user to easily compare the results of different simulations. A typical example of the final plot obtained from a simulation is shown in Fig. 2.7. Note that these links represent only the LOS links. The number indicated in the middle of each icon represents the level of that station. The station number is displayed next to each station, but might not be clearly visible for all stations, as the figure is very populated.

![Final plot of all the links for a network where all stations have a maximum transmit radius of 1 km](image)

**Figure 2.7:** Final plot of all the links for a network where all stations have a maximum transmit radius of 1 km

Figure 2.7 contains all the nodes specified in Table 2.1. Each station was set to have a maximum transmit radius of 1 km. Even though the idea is to have only 1 BS, multiple BSs were used to show that the network can be divided into four main clusters. These clusters should be connected to each other to create a meaningful network. Note that the network shown in this
Chapter 2 — Network Design

This network can be expanded further with the use of NLOS links and slightly longer links. The task of this program was only to automatically adjust the antennas to obtain the maximum number of LOS links. The MATLAB program does not take refraction or reflections into account and models only diffraction based on Fresnel Zones ($0.6F_1$). Radio propagation software must now be used to calculate the actual link budget of the entire network. The Radio Mobile radio propagation software is freely available and was thus used.

2.2.1.2 Radio Mobile

Radio Mobile propagation calculations are based on the US Institute for Telecommunications Science (ITS) propagation prediction model, better known as the Longley-Rice model. Radio Mobile can use various DEM formats such as SRTM, DTED, GTOPO30, GLOBE and BIL as source files to construct its terrain. The SRTM format is used, as it has the highest resolution and is the most accurate format. Roger Coudé, the designer of Radio Mobile, performed numerous tests to compare actual measured results to the predicted values obtained using his software. These tests were conducted in various different terrains. The terrain used in his sixth test resembles that of the terrain under study. His measured results are shown in Fig. 2.8, while the predicted values are shown in Fig. 2.9. The Radio Mobile predicted path loss was obtained using the 70% spot option, together with a rural hilly terrain with few trees. The results are quite similar, but the predicted values tend to be a bit pessimistic. These pessimistic predictions applied equally to all his tests. The Radio Mobile solution, therefore, can be used as the worst case scenario. This pessimistic edge can be used as a fading margin.

Radio Mobile allows the user to import the text file (RadioMobile.txt) created by the MATLAB program, which contains the adjusted GPS coordinates of all the points. This file also specifies...
the type of icon to use for each point and other display settings. Each station or node is referred to as a unit in the Radio Mobile environment. With all the units specified, the program needs some network settings before it can start analyzing the network. The network settings are best explained with reference to Fig. 2.10, which is an screen capture of the parameters and the system settings. The network name (net name) allows the user to identify different networks within the same area. Only one network is used in this thesis. According to [Cou03] the best solution for this problem will be obtained with the mode of variability set to 70% at the same spot. The terrain does not contain any forests and is not situated near to any big towns or cities. It is proposed by the software manual that the default value be used for the surface refractivity, while the ground conductivity and relative ground permittivity should be choosen according to Table 2.2. The surface of this area can be described as average ground.

**Figure 2.9:** UHF predicted values taken from [Cou03]

**Figure 2.10:** Screencapture of the network parameters and systems tabs
Chapter 2 — Network Design

Table 2.2: Proposed ground properties according to the Radio Mobile manual

<table>
<thead>
<tr>
<th>Ground attribute</th>
<th>Ground Conductivity</th>
<th>Relative Permittivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average ground</td>
<td>.005</td>
<td>15</td>
</tr>
<tr>
<td>Poor ground</td>
<td>.001</td>
<td>4</td>
</tr>
<tr>
<td>Good ground</td>
<td>.02</td>
<td>25</td>
</tr>
<tr>
<td>Fresh water</td>
<td>.01</td>
<td>25</td>
</tr>
<tr>
<td>Sea water</td>
<td>5</td>
<td>25</td>
</tr>
</tbody>
</table>

The system settings specify the main variables required to calculate the link budget of each link. These settings include the transmitter output power, the receiver sensitivity and the antenna gain, height and type (radiation pattern). Additional line and cable losses can also be added. Multiple systems can be created within the same network. This allows the user to change the settings of certain nodes, for example the antenna gain and type of a single node, which require specific attention. The settings used are shown in Fig. 2.10.

The topology setting specifies the way in which the different links are connected. These settings are based on the type of communication required and the communication protocol used. Each of the different topology settings has different types of sub nodes. The following topology settings exist:

- **Voice net** - This type of communication allows the user to set each unit as either a command node, a subordinate node or a rebroadcast node. Connections are required only between command posts and subordinate units, but not between subordinates. Rebroadcast units operate as repeater stations and can increase the communication range.

- **Data net, star topology** - With this topology each unit can be either a master node or a slave node. This topology should be used in polling type networks where the master station polls all slave units, with no links between the slave nodes. Note that no multi-hop is allowed in this type of network.

- **Data net, cluster** - With this topology each unit is set as either a node or a terminal. All nodes can retransmit messages and the number of rebroadcasts allowed can be limited if necessary. This topology is the best description of the desired network.

Each unit that is supposed to be part of the network must be added to the membership list in the network properties tab. Each member must specify the system it belongs to, as well as its antenna height, if this differs from the system height. The sub class of each unit must also be specified, depending on the network topology used.
2.2.1.3 Radio Mobile results

The previous section described the setup procedure required before the network analysis can begin. In this section the results obtained from five different tests will be discussed. The voice net topology was used to classify the type of network in the analyses, with each node set as a rebroadcast node. The voice net was used rather than the data net cluster, even though the data net cluster is a more accurate description of the network at hand. The reason for not using the data net cluster topology is because it tends to display too many connections, which makes it difficult to see the units. The voice net topology only requires that each node is linked via the best link. Note that the style properties for this network were set to display only links with relative power greater or equal to 3 dB. The five tests performed are:

- Test 1a - With this test the voice net topology was used with all the nodes set as rebroadcast nodes. Only one system, named RSB, is created with transmit power of 10 dBm, receiver sensitivity of -100 dBm and an omni directional antenna with gain of 0 dBi. The antenna height is set to 5 m. The line loss is set to 1 dB, with an additional cable loss of 1 dB/m. The results obtained for this test are shown in Fig. 2.11. This network contains two unconnected nodes, those of units 31 and 32, and five other clusters containing two stations or more. All the stations must be connected to create a meaningful network.

![Figure 2.11: Results for test 1a](image)
• Test 1b - This test is exactly the same as test 1a, but an additional three test points, denoted as TP1, TP2 and TP3, and a new system, that of RSB2, is added. RSB2 uses an antenna with a 2 dBi gain to try and increase the transmission distance. Only unit 33 (monitor point B02.04) is assigned to use RSB2 as its system. The three test points added are used to connect the different clusters. Higher gain antennas cannot be used to solve this problem because the connections between links are obstructed by mountains. These three additional stations would not perform any ADC monitoring, but would only operate as repeater stations. As described in Section 4.4, the cost of an RSB is not that high and, therefore, the addition of these three stations is a valid solution. Battery packs or solar panels can be used to supply these stations with power. The GPS coordinates of these three test points are given in Table 2.3. With the addition of these three test points and the RSB2 system, all the monitoring points can be accessed as shown in Fig. 2.12.

<table>
<thead>
<tr>
<th>Name</th>
<th>Latitude [S]</th>
<th>Longitude [E]</th>
<th>Elevation [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP1</td>
<td>33°36’6”</td>
<td>019°44’15.5”</td>
<td>892.4</td>
</tr>
<tr>
<td>TP2</td>
<td>33°37’45.4”</td>
<td>019°46’14.5”</td>
<td>989.6</td>
</tr>
<tr>
<td>TP3</td>
<td>33°36’42.9”</td>
<td>019°48’03.9”</td>
<td>1092.4</td>
</tr>
</tbody>
</table>

Table 2.3: GPS coordinates of the added test points

Figure 2.12: Results for test 1b

• Test 2a - The actual power transmitted by the RSB, with the transceiver set to the 10
dBm setting, can be as low as 6 dBm, as is explained in Chapter 4. A network should always be designed to be able to withstand the worst case scenario. This necessitates the use of test 2a, which is exactly the same as test 1a, but with the system transmit power reduced to 6 dBm. As expected, this results in the loss of numerous links. In total eight stations (29, 30, 31, 32, 36, 34, 52 and 51) and five clusters are not connected. The results obtained are shown in Fig. 2.13.

![Figure 2.13: Results for test 2a](image)

- **Test 2b** - The same approach as was used in test 1b is applied to test 2a, to try and connect the stations mentioned in test 2a. The same test points (TP1, TP2 and TP3) defined in test 1b are used, but the transmit power of the RSB2 system is reduced to 6 dBm. Units 54, 52, 51 and TP2 are assigned to use system RSB2. Notwithstanding all these adjustments, one station (34) and three clusters are still not connected.

- **Test 2c** - Test 2b is now extended by using another system, RSB3. With RSB3 the transmitted power remains 6 dBm, but the antenna gain is boosted to 4 dBi. The same network settings as defined in test 2b are used, with the addition of unit 18 being assigned to use RSB2 and units 34 and 33 assigned to use RSB3. These adjustments produce the fully connected network shown in Fig. 2.14. The network can better be described using the full three dimensional view of the network shown in Fig. 2.15. Note that the unit markers do not represent the actual antenna heights.
Figure 2.14: Results for test 2c

2.3 Summary

In this chapter the different aspects that have to be taken into consideration when calculating the link budget of a link, were discussed. The complete network desired was also analyzed using the Radio Mobile radio propagation prediction software. The radio settings of the proposed hardware designed in Chapter 4 were used to test whether the hardware design would provide sufficient radio coverage. The network shown in Fig. 2.15 can be obtained using the designed hardware in collaboration with omni directional antennas with gains of 0 dBi, 2 dBi and 4 dBi respectively. Note that the hardware design containing the completed RF modules must be used to ensure that the transmitted power is greater than 6 dBm. Also note that the links shown in Fig. 2.15 represent only the links that are greater than or equal to 3 dB. As explained in Section 2.2.1.3 the path loss values predicted by Radio Mobile also tend to be 10 dB higher than actual measured results. This 10 dB gain is used as a fading margin to allow for signal degradation.
Figure 2.15: A 3D representation of the results obtained from test 2c.
Chapter 3

Communication Strategy

Before any data communication can occur, some sort of communication strategy has to be established. The OSI (Open Systems Interconnection) seven layer model can be used to help with the protocol design. A schematic representation of this model is shown in Fig. 3.1. This model has been a very popular design tool since it was introduced in the late 1970s. The OSI model was initially developed in an attempt to standardize protocols. Each of the seven layers present in this model has a very specific task to perform. The seven layers are:

1. Physical layer - This layer represents the physical hardware with its interconnections. Modulation and demodulation of data is also part of this layer. Data in this level is studied on bit level.

2. Data link layer - This layer describes the basic transmission of data and error control. Data in this level is handled as frames.

3. Network layer - The network layer has to manage all the routing of data, including the addressing and congestion control. All the routing information is added to the packet headers and, therefore, all data in this layer is handled as completed packets.

4. Transport layer - The transport layer controls the reliability of a given link through flow control. Basically, this layer describes how techniques such as retransmission of data can be used to ensure successful data communication.

5. Session layer - This layer is used for dialogue control and token management. This layer is used to establish and terminate connections.

6. Presentation layer - This layer only works with the data portion of packets (no headers or error correcting bytes), and describes the syntax and semantics used. An example of a task performed by this layer is the way the GUI, which was programmed using object Pascal, communicates with the PIC, which was programmed using embedded C.
7. Application layer - The application layer is the highest layer in the OSI model and is responsible for data translation and interpretation. This layer is responsible for file transfer, formatting and terminal compatibility.

3.1 Physical layer

3.1.1 Hardware

The design of the hardware is explained in detail in Chapter 4. The motivation for using the implemented hardware components is also supplied in that chapter. Some relevant information regarding the hardware has not been mentioned, and will be discussed in this section. Two transceivers were used for data communication. Each transceiver is only capable of communicating in half-duplex mode. The addition of the second transceiver does not create a full-duplex link, because the two transceivers are not dedicated RX and TX channels. Even though the RSB is allowed to transmit and receive at the same time, it cannot be defined as a full-duplex network, but rather as a network with dual half-duplex channels.

The nRF905 transceiver uses constant length packets, with a maximum payload length of 32 bytes. The use of fixed length packets decreases the overall efficiency of the network, because the packet length will remain at maximum length, independent of the number of bytes that must be sent. Some scheme might be implemented where one transceiver can be used to send short control packets, while the other transceiver is used for data transfer. The short transition
3.1.2 Modulation

The nRF905 transceiver uses GFSK modulation, which is much more bandwidth efficient than ordinary FSK modulation. The only difference between FSK and GFSK is that GFSK uses a Gaussian filter to smooth the base band pulses before they enter the modulator. This filtering limits the spectral width of the signal. GFSK is used in most bluetooth devices. The signal is also Manchester encoded to ensure synchronization. Manchester encoding, also referred to as split-phase ([ZT02]), is a line code where every encoded bit has at least one transition per clock cycle. With this scheme the clock pulse can always be extracted from a received signal. An example of a Manchester encoded message is shown in Fig. 3.2. Taking the modulation and demodulation into account, the nRF905 transceiver delivers a final data rate of 50kbps.

3.2 Data link layer

3.2.1 Error control

An important factor to consider is that of channel noise. Numerous noise sources exist that can severely deteriorate the SNR of a signal. These interferences are caused by both external and internal sources. The internal noise consists of thermal noise, which is caused by random electron movement, as well as harmonic noise, which is caused by the RF circuitry ([Tec06]). External sources include radiation from the sun, lightning, static and harmonics generated by other electrical appliances. Multi path transmission also plays an enormous role in the received SNR of a signal. Other sources of signal deterioration also exist.

To combat the influence of noise, some sort of error handling has to be implemented. If no error
detection is used, a station would be unsure of the reliability of the data received. All error detection and error correction schemes require the addition of extra redundant data. Redundancy decreases the efficiency of a network, and is defined by eq. 3.1. Note that the total number of bits in this equation includes retransmitted data. Two basic protocol types that can be used to implement error control are automatic repeat-request (ARQ) and forward error correction (FEC).

\[
R = \frac{\text{Totalbits}}{\text{Informationbits}}; \quad \text{and} \quad \eta = \frac{1}{R} \tag{3.1}
\]

### 3.2.2 Automatic repeat-request (ARQ)

With the ARQ protocol, error detection codes are used to determine whether received data is correct or not. This protocol is completely embedded in the data link layer, and uses acknowledgments (ACK) and timeouts (TTL) to achieve reliable data transmission. ACK messages are used by a receiving station to confirm reception of valid data. Timeouts are used by transmitting stations as an indication of how long they should wait for an ACK message, before assuming that the packet got lost. When a timeout occurs a packet is either retransmitted until an ACK is received, or until the maximum allowable number of retransmissions has been reached. Three commonly used ARQ protocols are “stop and wait”, “go back N” and “selective repeat” protocols. Hybrid ARQ (HARQ) protocols also exist, but are more complicated to implement.

#### 3.2.2.1 Stop and wait ARQ

A schematic explanation of the pure stop and wait protocol is shown in Fig. 3.3. The NAK (not acknowledged) command is used to report received packets which did not pass the CRC check. On reception of the NAK command, the initial source station would retransmit the message. Packets can also be lost completely, in which case the receiver would not ever detect them. To prevent a station from waiting for ever for an ACK message, TTL timers are introduced. Whenever a packet is transmitted, a timer is set for that packet. If an ACK is not received before the timer expires, the packet is retransmitted. The timer must allow the ACK message enough time to be received. This time should include the time needed to transmit both the message (MSG) and the ACK. It should also include processing time and an additional short period of time to compensate for back-off periods. Note that with this protocol a new packet cannot be transmitted before the ACK of a previous message is received.

When TTL timers are used, the NAK messages can be removed if latency is not that important. If NAK messages are used, the source station would start retransmitting on reception of the NAK, while if these messages are not used, the station would have to wait for the TTL timer to expire. This can be explained by reference to Fig. 3.3. If NAK messages are used, the source station would realize that an error has occurred with packet 1 at time \( t_4 \), while it would only realize this at time \( t_{TTL2} \), if no NAK messages are used. An advantage of excluding this NAK
message is that the total traffic density is reduced. Less traffic will decrease the probability of error caused due to collisions.

When using retransmission, a station should always know which packet to retransmit. Packet numbers should therefore be added to each packet to allow the station to identify the problem packets. This packet number should be added to both the MSG and ACK packets. To enable a station to retransmit a packet again at a later stage, all transmitted packets have to be stored in memory. A packet can only be removed from memory if an ACK is received or if the maximum number of retransmissions allowed per packet is reached.

### 3.2.2.2 Go back N ARQ

The stop and wait protocol is inefficient if the propagation delay is greater than the packet transmission time. With the “go back N” protocol new packets can be sent before all earlier packets have been acknowledged. The number of packets that can be sent without acknowledgment is referred to as a window. This window can move forward as ACKs are received for earlier messages. The receiver still acknowledges all packets individually. The receiver can only receive packets in sequential order. Once a packet is lost or in error, the data will no longer be sequential, and the receiver will stop sending ACKs. After the TTL of the window has timed out, the window will restart transmission from where the last ACK was received.

The advantage of this protocol is that multiple packets can be transmitted at the same time, increasing the utilization of the network. A disadvantage is that all packets that were transmitted after an error packet will have to be retransmitted. It is, therefore, possible that a complete window need to be retransmitted, due to one error. This protocol is intended for networks with very long propagation times and duplex channels. This protocol is not optimal for the network under discussion, or the chosen hardware.
3.2.2.3 Selective repeat ARQ

This protocol is quite similar to that of “go back N”, except that data does not have to be received sequentially. The receiving station automatically rearranges the packets according to their sequence numbers. The window’s lower boundary is still determined by the packet with the lowest sequence number which has not yet received an ACK.

3.2.2.4 Error detection

Numerous error detection schemes exist, but the nRF905 transceiver has a built in CRC (cyclical redundancy check) generator and detector, which can be used for error detection. CRC is a very powerful detection scheme based on polynomial mathematics. The CRC function of the nRF905 chip can be either disabled or enabled in CRC-8 or CRC-16 mode. According to [RW06], a 16-bit checksum, such as CRC-16, catches:

- All single and double errors
- All errors with an odd number of bits
- All burst errors of length 16 or less
- 99.997% of all 17 bit errors
- 99.998% of all errors \( \geq 18 \) bits

3.2.3 Forward error correction (FEC)

Error detection codes can only detect errors, but not correct them. Error correcting codes also exist, but at the expense of even more redundant data. FEC requires the use of error correction codes. FEC codes can be classified as either block codes or convolution codes. Convolution codes are used mainly for direct data streaming and require quite a large amount of processing power to implement. This type of code is typically used to enhance the quality of digital radio, mobile phones, satellite links and other Bluetooth implementations. Numerous block codes exist, with Reed-Solomon codes, Golay codes, BCH codes and Hamming codes being the most popular. Block codes requires less computational power and are easier to implement than FEC convolutional codes.

The problem with error correction codes is that they have a high level of redundancy compared to error detection codes. The number of errors that can be corrected is also limited and is a function of the number of redundant bits. Typical BCH codes that might be used are the \( BCH(255, 239, 2) \), \( BCH(255, 231, 3) \) or \( BCH(255, 223, 4) \) code. The \( B(n, k, t) \) notation used describes the level of redundancy used and the number of errors that can be corrected. With
this notation $n$ denotes the total number of bits in a packet, $k$ denotes the number of information bits and $t$ denotes the number of errors that can be fixed. The optimal code must be chosen depending on the link quality.

3.2.4 Implemented error control

The problem with the use of FEC, is that the redundant bytes have to be added to the payload. With the maximum number of payload bytes limited to 32, the addition of an error correction code, such as the BCH(255,231,3) code, will reduce the number of bytes available to only 28 (12.5% reduction). Note that the packet header still has to be added to the available data portion.

Latency is not that important for this network and the available CRC hardware makes it very easy to implement an ARQ type system. The congestion of the required channel is also very low, resulting in retransmissions not having such a big influence on the network. An ARQ protocol with CRC error detection, generated by the hardware of the nRF905 transceiver is, therefore, used to implement error control.

With the pure stop and wait protocol a new message cannot be sent before a previous message has been acknowledged. This adds unnecessary idle times. With the type of network designed, numerous stations which all want to communicate with the same station can be present. This network is also not used for bulk data transfer. Each packet can be interpreted as an individual entity. It is not a necessity that packets are received in sequential order to be able to interpret the data. New data can therefore be transmitted before a previous packet has been received. The protocol used is a hybrid type of selective repeat protocol. Sequence is not an issue and packets are numbered only to identify which packets should be retransmitted. The window size is also assumed to be equal to the TX buffer capacity.

The protocol used can be summarized as follows: Any packet that must be sent from a station, is transmitted from that station, independent of previously sent items. This message is added to an ACK stack, together with a TTL timer and a retransmission counter. Each packet contains a CRC error detection code which must be used by the receiving station to test for validity. The receiving station replies with an ACK if the message is correct, otherwise it simply ignores the packet. If an ACK is received at the source station, the corresponding packet is removed from the ACK stack. If a TTL timer expires before a packet is removed, that packet is retransmitted, if its retransmission counter is less than the maximum allowable number of retransmissions, otherwise the packet is dropped.
3.2.5 Basic transmission protocol

The way communication is initialized and the way the communication channel is accessed are very important factors in network design. Channel access can be divided into circuit mode and channelization methods and packet mode methods. Circuit mode and channelization methods describe the way a station can physically access the channel. This includes the use of schemes such as frequency division multiple access (FDMA), time-division multiple access (TDMA) and spread spectrum multiple access (SSMA). None of these techniques were used in this thesis. Packet method modes describe a more software defined approach to channel access. These methods can be subdivided into two main categories: collision free protocols and contention protocols. With collision free protocols only one station is in control and it schedules which stations can gain access to the channel. The advantage of this scheduling procedure is that it provides a collision free protocol, as only one station would have channel access at a time. This protocol is very stable and efficient under heavy loading, but tends to be very slow in low data traffic networks. Examples of collision free protocols are: round-robin polling (RRP), bit-map protocol, adaptive tree walk protocol, token ring, token bus and distributed queue dual bus (DQDB) (see [RW06]).

With contention protocols each station would try and access the communication channel automatically when it has data to send. The way in which a station decides when to access the channel differs for each protocol. Contention protocols are faster and more efficient than collision free protocols in low data traffic networks, but tend to be very inefficient under heavy loading. Examples of contention protocols are: ALOHA, slotted ALOHA, carrier sense multiple access CSMA, carrier sense multiple access with collision detection (CSMA/CD) and carrier sense multiple access with collision avoidance (CSMA/CA). The CSMA protocol is used in this network. The motivation for using CSMA can be found in Chapter 7.

3.2.5.1 Carrier sense multiple access (CSMA)

CSMA is a media access control (MAC) protocol which resides completely within the data link layer. With CSMA a station listens (also referred to as sniffing) to establish whether another station is already transmitting data before it tries to transmit its own data. If a carrier is detected the station would go into back-off mode and come back later to try and transmit. If no carrier is detected, the station would assume that the channel is clear and would immediately try and transmit its own data.

The ARQ protocol described earlier is used in conjunction with the CSMA protocol to ensure error free transmission. If a valid ACK message is not received before a timeout occurs, the station would retransmit the message. Note that two ACK messages exist. An ACK2 message is sent back by a destination station on reception of a valid packet, while an ACK1 message is sent back by any station forwarding the data. The use of ACK packets can best be described
with reference to Fig. 3.4. RSB1 is sending data to RSB4 via RSB2 and RSB3. When data is transmitted from RSB1, the packet would be stored in the RSB1 ACK stack. This entry has two flags indicating that it is still waiting for an ACK1 and an ACK2 message. The RSB2 station would send an ACK1 message back to RSB1 on reception of the data, but it would also transmit the message to RSB3 and add it to its own ACK stack. RSB2 would not require an ACK2 message to confirm reception at the destination, because it did not originally send the message. When the ACK1 message reaches RSB1, it would clear the ACK1 flag to indicate that the message is successfully on its way to RSB4. If the ACK1 message does not reach RSB1 before a timeout, it would retransmit the message. Note that the message is not removed from the ACK stack when the ACK1 message is received. The same process would be repeated when the data reaches RSB3, but the RSB2 station would completely remove the packet entry from its ACK stack if it receives a valid ACK1 message from RSB3. When RSB4 receives the data it sends an ACK2 message back to the source station. Note that RSB3 has not yet received an ACK1 message from RSB4, but will remove the stack entry if it receives the ACK2 message. RSB3 and RSB2 would simply forward this ACK2 packet back to the source station without any further stack additions. If RSB1 receives the ACK2 message it would also remove the entry from its own ACK stack. As explained in Section 3.3, all data transfers go via the BS. The source station always knows how many hops are required to reach the BS. If the TTL for a single hop transmission is $x$ seconds, and $h$ hops are required to reach the BS, then the ACK2 TTL will be set to a value greater than $hx$ seconds.

The ACK1 is used to quickly retransmit lost packets between the forwarding stations, while the ACK2 message is used to retransmit a packet completely. A problem with this protocol is that no form of acknowledgment is used to confirm the reception of ACK packets. ACK messages can also not be retransmitted. Once the ACK2 message has been transmitted from RSB4 no form of error control is used until it reaches RSB1. In noisy networks this could cause major problems.

3.3 Network layer

3.3.1 Routing

An ad-hoc network is an infrastructureless network. This means that no predefined routers and gateways are set up for the network. Each station must perform the tasks of both a monitoring
station and a router. The network topology and infrastructure of this type of network can change during operation. As explained by [Nic06], the type of data acquisition network under study can be described as a nomadic network rather than a mobile network. This assumption implies that even though the stations are mobile, they would not move very quickly or frequently. Due to this nomadic nature the routing tables do not have to be updated as often as they would have to be with mobile networks. According to [Ha03], ad-hoc routing protocols can be divided into two classes: table-driven and on-demand routing. The two classes differ from each other based on the frequency and way they discover new routes. With table-driven routing protocols a table containing the routes to every station in the network should be held at each station. This requires constant updating. On-demand routing protocols do not keep routing tables, but rather create routes when desired by a source station. Commonly used routing protocols are listed below, but not discussed. For more information on the difference between these protocols, refer to [Nic06] and [Ha03].

**Commonly used table-driven protocols** - Destination-sequenced distance-vector routing (DSDV), wireless routing protocol (WRP), global state routing (GSR), fisheye state routing (FSR), hierarchical state routing (HSR), zone-based hierarchical link state routing protocol (ZHLS) and cluster-head gateway switch routing protocol (CGSR).

**Commonly used on-demand routing protocols** - Temporally ordered routing algorithm (TORA), dynamic source routing protocol (DSR), cluster-based routing protocol (CBRP), ad hoc on-demand distance-vector routing (AODV), signal stability-based adaptive routing (SSA), associativity-based routing (ABR), optimized link state routing (OLSR), zone routing protocol (ZRP) and virtual subnets protocol (VSP).

Table-driven routing protocols are best suited for networks with high mobility. The frequently updated routing tables allow stations to quickly adapt to new topologies. With nomadic networks these protocols tend to be less efficient. The network topology does not change constantly and, therefore, the constant routing update messages become unnecessary overhead.

With on-demand protocols, like DSR, previous routes are kept in cache for quick access. If the network topology does not change, route exploration to a specific node would only have to occur at the beginning of the first transmission to that node. Consequent packets would detect an existing route in the cache memory, and would not have to look for a route to the destination. A problem with DSR is that the complete route must be added to the packet header. The number of payload bytes available are limited, which makes the addition of routing information problematic. With the hardware used, all packet lengths are constant. This removes the option of sending short route exploration packets. Long packet headers are also a limiting factor.

Some hybrid protocol should be designed to take all these design constraints into account. The constraints can be summarized as follow:

- Constant packet size - Packet length is constant, which prevents the use of short routing
Figure 3.5: Example network layout of an ad-hoc network

exploration messages.

- Limited payload size - The payload length is fairly short, which prevents the use of long routing headers.
- Nomadic network - This network property makes it unnecessary to frequently update the routing tables.

3.3.1.1 Hybrid routing protocol design

The main reason for designing this network is to log all the data measured at the numerous stations. Data would seldom be sent from one station to another, where none of these stations is the BS. The BS is, therefore, the most important station in the network and must always be active and reachable. It can be stated with complete certainty that the BS is the centre of the network, which makes it even more logical to make this station the centre node. All stations must be linked in such a manner that the shortest possible link to the BS is used. This can be presented graphically as in Fig. 3.5.

This network resembles that of a hierarchical topology. This topology permits the use of a hybrid protocol based on the table-driven hierarchical state routing (HSR) protocol. The HSR protocol is adjusted to make the BS the center of the network. The protocol is reduced further by creating a full routing table only at the BS. All the other stations keep a list of only their children, their parent station and their access point (AP) station. The AP station is the same as the BS, but this term is used to allow the use of multiple base stations, in which case each
BS would merely be an AP to the bigger network. The motivation for keeping a complete routing table only at the BS is as follows:

- All stations would transmit their measured data to the BS, which can always be reached by simply sending data to the parent station until the BS is reached.
- It is very simple for a BS to send data to a non-BS, as the BS has a complete routing table. See Section 3.3.2 for more detail on addressing.
- If a non-BS wants to send data to another non-BS, it simply has to send the data via the BS. In this case the BS would perform the same task as a gateway in the normal HSR protocol. This might seem like a redundant loop, but this type of non-BS to non-BS communication is very rare.
- The total amount of traffic caused by sending route update packets, is reduced enormously by creating a route table only at the BS. This solves the problem of too much overhead caused by table-driven protocols.

With this protocol each station can be in one of 2 states: routed or unrouted. An unrouted station is not part of the network and is denoted level 0. If a station’s level is greater than 0, it is a routed station. A station is set as a BS by setting its level to 1. All routing tables are created from the BS downwards. The BS first searches for all stations within transmit radius of itself. These stations are called children (level 2). After the BS has finished searching for children, it would start searching for grandchildren. This is done by allowing each child sequentially to look for children of their own. After all the BS’s children have finished searching for children, the BS allows each grandchild sequentially to search for their own children. This process will continue until no new children can be found. Every station then transmits its child list to the BS, which compiles a complete list. The sequential order in which stations are added ensures that each station will always have the shortest route to the BS. This route setup procedure is referred to as scanning. Scanning must be repeated periodically to ensure that all links stay up to date. The inter scan time can be set depending on the level of mobility of the network. During this table setup procedure some common rules are applied. A station would only change its level if a shorter route is available to the BS, or if the existing route has not been confirmed for a very long time, in which case the station’s level will be reset to 0.

A more detailed description of the scanning process is given in Section 3.7. The adjusted HSR protocol discussed thus far does not specify how stations are addressed or how the routing header is constructed.

### 3.3.2 Addressing

Each station must be allocated a unique name to enable the server to identify the source of each received measurement. This network is intended mainly for monitoring of bore holes, fountains
and dams. The monitoring points are currently referred to using a 4 letter code, which was assigned to them by the Department of Water Affairs. This code includes farm names, bore hole numbers and other descriptive information. To simplify this process each station is given the corresponding 4 letter code. This code is used as the board name. Every station also has a level, which indicates how far this station is from the BS. Even though this 4 letter code can be used to uniquely identify each station, it gives no routing information at all. The maximum payload length is also too short to add long routing headers. Routing information is not needed when data is sent towards the BS, but is a necessity when data is sent from the BS. The only way to send data without routing information is by either having completed routing tables available at each station, or by flooding the network. Routing tables are not present at each station and flooding severely aggravates network traffic congestion. To solve this problem a dynamic routing code is introduced.

This 16 bit code is assigned to a station during the scan procedure and is bound to change every time a change in the network topology occurs. Network scanning is performed sequentially by traversing from level 1 through to the highest level, as explained in Section 3.3.1.1. A BS is always assigned code 0. After the BS has finished scanning for children of its own, it determines the number of bits required to count to that number. If 8 children were found, 4 bits (1000_b) would be needed. Each child would then be issued with a code corresponding to its number in the child list. Each child would also be informed that 4 bits were used to identify all the children in that level. When a child has finished searching for children the same procedure is used, but the child’s own code is appended to the end of the existing code. For instance if the third child in the previous example had five children, it would require 3 bits (101_b) to represent each of them. The total code length for each of these five level 3 stations would, therefore, be 7 bits. The fifth child’s code would be 1010011, with the level 2 station’s code being 0011_b and the level 3 station’s code being 101_b. Each of the level 2 station’s children would then be informed that the current number of bits used in their code is 7 bits. The length of the code is added to make sure that the station knows how many padding zeros must be added to the parent station’s code. Note that the complete code is 16 bits long. The routing codes for the network topology shown in Fig. 3.5 is given in Table 3.1. Note how stations on the same level would not necessarily have equal length codes.

When a station sends data to the BS, it adds its own code to the message. The BS can be reached by traversing through parent stations until the BS is reached. When a message or an ACK is sent from the BS to a higher level station, the code of the destination station is added to the packet. On reception of a packet a station would extract this code and compare it with its own code. If the code length of this station is k bits long, the destination station would be a descendant of this station if the last k bits are exactly the same. This test is used to determine whether a packet should be forwarded or not. This test works for transmission in both directions, seeing that it merely confirms whether the code belongs to a descendant station or not. A quick and easy way to test this is to XOR the received code with your own code. If the last k bits is 0, the station is a descendant. By employing this scheme the routing overhead
### Table 3.1: Routing codes for the network shown in fig. 3.5

<table>
<thead>
<tr>
<th>Station name</th>
<th>Station Level</th>
<th>Parent station</th>
<th>Binary code</th>
<th>Decimal code</th>
<th>code length</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS</td>
<td>1</td>
<td>BS</td>
<td>00000000000000000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>BS</td>
<td>00000000000000001</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>BS</td>
<td>00000000000000001</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>BS</td>
<td>00000000000000011</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>BS</td>
<td>00000000000000100</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>A</td>
<td>00000000000001001</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>A</td>
<td>00000000000010001</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>A</td>
<td>00000000000110101</td>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>H</td>
<td>3</td>
<td>BS</td>
<td>00000000001010010</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>I</td>
<td>3</td>
<td>C</td>
<td>00000000000110110</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>J</td>
<td>3</td>
<td>D</td>
<td>00000000001010000</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>K</td>
<td>3</td>
<td>D</td>
<td>00000000001010010</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>L</td>
<td>4</td>
<td>E</td>
<td>00000000000110100</td>
<td>26</td>
<td>5</td>
</tr>
<tr>
<td>M</td>
<td>4</td>
<td>E</td>
<td>00000000001110000</td>
<td>27</td>
<td>5</td>
</tr>
<tr>
<td>N</td>
<td>4</td>
<td>H</td>
<td>00000000001010001</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>O</td>
<td>4</td>
<td>I</td>
<td>00000000001110010</td>
<td>21</td>
<td>8</td>
</tr>
<tr>
<td>P</td>
<td>4</td>
<td>J</td>
<td>00000000001110010</td>
<td>22</td>
<td>8</td>
</tr>
<tr>
<td>Q</td>
<td>5</td>
<td>M</td>
<td>00000000001110010</td>
<td>58</td>
<td>6</td>
</tr>
<tr>
<td>R</td>
<td>5</td>
<td>N</td>
<td>00000000001110010</td>
<td>59</td>
<td>6</td>
</tr>
<tr>
<td>S</td>
<td>5</td>
<td>O</td>
<td>00000000001110010</td>
<td>108</td>
<td>7</td>
</tr>
<tr>
<td>T</td>
<td>5</td>
<td>P</td>
<td>00000000001110100</td>
<td>187</td>
<td>8</td>
</tr>
<tr>
<td>U</td>
<td>6</td>
<td>Q</td>
<td>00000000011110010</td>
<td>236</td>
<td>8</td>
</tr>
<tr>
<td>V</td>
<td>6</td>
<td>S</td>
<td>00000000011110100</td>
<td>379</td>
<td>9</td>
</tr>
<tr>
<td>W</td>
<td>6</td>
<td>T</td>
<td>00000000011110101</td>
<td>492</td>
<td>9</td>
</tr>
</tbody>
</table>

is always 2 bytes, independent of the number of hops required.

### 3.3.3 Packet header construction

In order to send meaningful data, some useful header information must be added to packets. When a packet is received a station should be able to identify the type of packet. Five basic types are used: message (MSG), command (CMD), ACK1, ACK2 and performance (PERF). A three bit type field is used to distinct between the different types. A station should also be able to identify exactly what it should do if it receives a CMD packet. An extra 5 bit command (cmd) field is added for sub class descriptions. The ARQ protocol requires the use of packet numbers (PN) in order to uniquely identify packets for retransmission. A single PN byte is added to the header for this purpose. For routing purposes the station should also add the
routing code described in Section 3.3.2 to the header. A two byte \textbf{CODE} field is added to the header. The routing code of the stations can change every time a new scan sequence is performed and can, therefore, not be used to describe the source or destination stations. The ID name and level of both the source and destination station are added to uniquely identify each station. Even though the ID name of each station is 4 bytes long, adding the complete name is justified, as this decreases complexity. This network is not intended for high data traffic applications, which further justifies the use of 5 identification bytes per station. These identification fields are denoted as \textbf{SID} (source ID), \textbf{SLVL} (source level), \textbf{DID} (destination ID) and \textbf{DLVL} (destination level). The last field required is that of a data byte (\textbf{DB}) counter. This field is used to indicate the number of data bytes used to store data.

The complete layout of the data packet is shown in Fig. 3.6. The command set used for communication is part of the presentation layer and is explained in Section 3.5.1.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure3.6}
\caption{Packet header layout}
\end{figure}

\section*{3.4 Transport and session layer}

It is not a necessity that all layers have to be implemented in a final network protocol. The transport and session layers are often used in TCP/IP type networks, but are not implemented in this network.

\section*{3.5 Presentation layer}

The presentation layer is responsible for specifying the exact format, syntax and semantics of the transmitted information. This layer allows different applications to gain access to the same network based on a standard interface. The RF interface will be discussed in this section. Even though the serial command set has nothing to do with the RF interface, it also provides a platform for an application interface and should, therefore, also be mentioned. The complete serial command set is given in Table 6.1.

\subsection*{3.5.1 RF command set}

The RF command set refers to all commands that can be used in the packet headers. A complete list is shown in Table 3.2. Note that the \textbf{EXTENDED} command allows the use of an additional 8 bits for future additions. All the \textit{cmd} commands are used in collaboration with the CMD
type command, except for the PERFORMANCE command, which is used together with the PERF type command. The addition of the PERFORMANCE cmd, allows for easy recognition of performance ACK packets. These PERFORMANCE packets must be clearly visible to know which ACKs to extract and send to the server for logging.

Table 3.2: RF commandset

The MSG type packet is used to send measured ADC data to the server. When this type of packet is sent, the data in the cmd field is of no importance and could be set to anything. All the RSB ADC values are not sent to the server when a packet is transmitted, but only those values that have changed by more than 10%. The first 2 bytes of the data block indicate which ADC ports’ data is contained within the sent data. Each bit within this first 2 bytes points to a corresponding ADC port.

### 3.6 Application layer

This layer is not the end user interface (GUI), but provides services to the user-defined application. This layer is the link between the server and the presentation layer. The application layer is responsible for transferring the data extracted from the presentation layer to the server and presenting it in a neat, easily accessible way. In a way the serial command set is a part of the application layer rather than the presentation layer.
3.7 Scanning procedure

The basic scanning procedure was discussed in the previous sections, but a complete description is required to understand the complete routing process. The scanning process can be divided into three categories: scanning for new children (orphans), confirming old children and searching for grandchildren. The three processes will be described individually, before explaining the full sequence.

3.7.1 Child list

Before the different scan sequences can be explained, knowledge of the layout of the child list is required. Each entry in the child list spans 8 bytes and contains 7 fields. Fig. 3.7 shows the layout of a child entry. The use of each field can be explained as:

- **ID (Name)** - This field contains the ID name of the child.
- **Level** - This field contains the level of the child.
- **Active** - The Active field is used as a flag indicating whether all the descendants of this child have finished scanning for children. If this flag is set, a grandparent scan is still in progress in this part of the network.
- **IsParent** - This flag indicates whether this child is a parent or not.
- **TheScanForChild flag** is used to indicate which child is currently scanning for descendants.
- **NumChild** - How many children does this child have.
- **CODE** - The CODE field contains the routing code of that child, which is determined by the parent station as explained in Section 3.3.2.

![Figure 3.7: Layout of the child entry in the child list](image)

3.7.2 Scanning for orphans

The basic flowchart for finding orphans is shown in Fig. 3.8. Note that the corresponding embedded code procedure names are written at the entrance and exit ports in the figure. The basic packet layout is also displayed to the right of each link (format used is `Type|cmd|SID|SLVL|DID|`
In this figure RSB1 is the station scanning for orphans. RSB1 starts the scan procedure by transmitting a scan beacon. This CMD packet contains the SCANNET cmd with all the source station’s information, but with the DLVL set to 0. The DID is not important and can be set to anything. Note that a station does not necessarily have to be of level 0 to qualify as an orphan. A station would also reply to a SCANNET cmd if this new route will supply a link with fewer hops to the BS. This would happen if \((SLVL - 2) \leq LEVEL\), with LEVEL being the station’s current level. On reception of a SCANNET packet, RSB2 would reply with an ACK2 message if its own level is 0 or if \((SLVL - 2) \leq LEVEL\). The station would add its own information in the SID and SLVL fields to enable the parent scanning for orphans to identify which station has replied.

Figure 3.8: The basic scan procedure

If a valid ACK2 for this SCANNET packet is received, RSB1 would send a SetLVL cmd to the station which replied. This packet would contain the destination station’s new level in the first byte of the payload, as well as the 4 byte name of this station’s AP. When RSB2 receives this packet, it would set its own level to that of byte 1 and also copy the name of the AP to its memory, before sending an ACK2 message. RSB1 would add this station to its child list (see Section 3.7.1) on reception of this ACK2 message. Only the Name, Level and Active fields of the
child entry will be set. The other fields will be set during the grandparent scan. On reception of a valid SetLVL ACK2 packet, RSB1 will start a new orphan scan to search for more children. If the ACK2 for the SetLVL command is not received correctly then, after the maximum allowed retransmissions, RSB1 will start a new orphan scan (SCANNET). If an ACK2 for a SCANNET command is not received after the maximum allowed retransmissions, the “scan for orphan” process is ended.

### 3.7.3 Confirming old children

This process is managed by extracting entries from the child list. If there are no entries in the child list (no children) this procedure would not be executed. This scan is performed sequentially, starting with the first entry in the list. The parent station extracts the child’s name and level from the child list and copies it to the DID and DLVL fields. A SCANNET command is then sent to this child, which would reply with an ACK2 msg. If an valid ACK is received the Active flag (child entry) of that child will be set and the next child in the list would be prompted for confirmation. If an ACK is not received, that specific child will be deleted from the child list and the next child will be tested. This process is stopped when the end of the child list is reached.

### 3.7.4 Searching for grandchildren

With the scan procedures explained thus far the BS would only be able to detect level 2 stations. To expand the network the grandparent scan (GPS) is introduced. GPS works almost the same as the child confirmation procedure, with only a slight difference. With GPS all linked stations are sequentially allowed to scan for children, until no more orphan stations can be found.

To explain the GPS procedure, we revisit Fig. 3.5. The BS is in control of the main GPS. GPS is initiated after a normal scan for children has finished. After the normal scan procedure the BS knows how many children it has and assigns each child its routing code. This code is added to the child list. The BS starts the GPS by sending a SCANGP cmd to its first child (A). The routing code for station A is added to the packet header, from where station A can extract it. The routing code is only used for forwarding packets and is not needed for single hop communication. The SCANGP packet also contains numerous other settings which are stored in the information section of the packet. The settings sent are:

- **GPScanDepth** - This variable is added to the first byte of the information section and indicates at which level the station should perform a scan.
- **Length of route code** - This second byte gives the number of bits used to compile the route code for the intended station.
- **Time and date** - The time and date, at which the GPSCAN packet is sent, is added
to the packet and is used by the receiving station to synchronize its clock. Bytes 2 - 9 ($DATA[2..9]$) are used for this timestamp.

• Scan settings - The inter arrival time of the RTCC service routine is added to byte 10, the ScanPeriod to byte 11 and the DropPeriod to byte 12. These settings are sent to ensure that the scanning process is synchronized. The ScanPeriod can be used in collaboration with the RTCC interval to determine when the next scan would occur, while the DropPeriod is used to determine when a station should assume that it is no longer part of the network and should reset its level to 0. These parameters are explained in more detail in Chapter 5.

On reception of this SCANGP cmd station A would first set all its settings, which can be extracted from the information bytes. Note that these settings would only be changed if the scan depth is set to 2, indicating that a level 2 station must perform a scan. If station A already has children it would first confirm that all its children are still present before it would start searching for orphans. After finishing the scan procedure it would send a FINSCAN cmd to the BS to indicate that it has finished scanning. It would also add the number of children found to the information bytes. When the BS receives this FINSCAN packet it would add the number of children found to this child’s child list entry. If it has found any children the IsParent and Active flags will also be set. The same process will be repeated with all the level 2 stations (B,C and D).

After finishing the check on scanning of all the level 2 stations, the BS would check if any of the Active bits of children in the child list are still set. If any flag is still set, the BS would increase the GPScanDepth variable and send a GPSCAN packet to the first child with a set Active flag. In this case it would be station A. When A receives the GPSCAN packet it will detect that the GPScanDepth value is higher than its own level, thus requesting its own children to perform a scan. Station A will now react exactly the same as the BS did in the previous scan. It will assign a route code to each of its children (E,F and G) and inform them sequentially to perform a child scan of their own. Note that a station would not be informed to start a scan before a FINSCAN has been received from the previous station. Following the same procedure as above, station E would reply with a FINSCAN packet indicating that it has found 2 children of its own (L and M). Neither station F or G will find any children and will reply with a FINSCAN packet informing station A that it did not find any children. Station E’s Active flag will be set, while F and G’s flags will be cleared. Station A will now send a FINSCAN packet to the BS with the number of children found set to 1 to indicate that children have been found. The BS does not keep track of the number of grandchildren and the 1 is used only to indicate that station A is still active. If no children were found the number of children would have been set to 0 and the Active flag would have been cleared. The same procedure is now followed with B, C and D.

After finishing this level 3 grandparent scan, the BS would once again test whether any of its children were still active. The BS would detect that all its stations are still active and would
increase its GSPanDepth before sending a GPSCAN packet to station A. Station A would detect that only station E is still active and would send a SCANGP packet to that station. Station E would now perform exactly the same tasks as have been described in the previous paragraph. This scanning process will continue until all the BS’s children’s Active flags are cleared.

It must be remembered that every SCANGP and FINSCAN packet requires a corresponding ACK message, although this was never mentioned during the explanation.

### 3.7.5 Complete scan sequence

All the different scanning processes have now been explained and the complete scanning sequence can, therefore, be considered. The BS will initiate the full scanning sequence by confirming all old children. Once it has been confirmed which children is still present, the BS would start searching for orphans. After finishing the orphan scan a GPS will be started. This scan will continue until no child’s Active flag is set. A complete flow chart of the scanning process is shown in Fig. 3.9. Note that the GUI is used to set how frequently this scan procedure should be executed. The higher the mobility of the stations, the shorter the inter scan times should be. At the end of each scanning sequence, the BS requests a list of all the station’s children. These lists are sent to the server where the data is processed and a schematic layout of the network is generated.

### 3.8 Summary

This chapter described all the different protocols and techniques used to ensure error free communication and can be summarized as in Table 3.3.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency band</td>
<td>433MHz ISM band</td>
</tr>
<tr>
<td>Data rate</td>
<td>50 kbps</td>
</tr>
<tr>
<td>Data encoding</td>
<td>Manchester encoding</td>
</tr>
<tr>
<td>Modulation</td>
<td>GFSK</td>
</tr>
<tr>
<td>Communication type</td>
<td>Half-duplex (x2)</td>
</tr>
<tr>
<td>Error detection</td>
<td>CRC</td>
</tr>
<tr>
<td>Communication protocol</td>
<td>ARQ</td>
</tr>
<tr>
<td>MAC protocol</td>
<td>CSMA</td>
</tr>
<tr>
<td>Routing protocol</td>
<td>Adapted hierarchical state routing (HSR)</td>
</tr>
</tbody>
</table>

Table 3.3: Summary of the network
Figure 3.9: Flowchart of the complete scan sequence
Chapter 4

Hardware Design

In this chapter the design of the hardware will be discussed, together with the layout of the PCB. The different components considered will be mentioned and a motivation will be supplied for choosing each chip used. The basic function of each component will also be stated briefly. The main aim of the project is to minimize the cost of the components. The basic blocks needed for the RSB are shown in Fig. 4.1. The most important blocks are those of the microprocessor and the RF link. The initial design included an ethernet interface, which was intended to be used to connect this network to a WiMax backbone (IEEE 802.16 standard). One example circuit with this interface was designed, but the interface was never implemented, because no WiMax hardware could be obtained. The board containing the ethernet interface is referred to as the access point board (APB). The basic block diagram of this station is shown in Fig. 4.2. The only difference between the RSB and the APB is the replacement of the ADC port with the ethernet interface. The APB was not intended for data acquisition, but merely to serve as a gateway between the ad-hoc network operating at 433 MHz and the WiMax backbone operating at 2.5 GHz.

![RSB Hardware Block Diagram](image)

**Figure 4.1:** RSB hardware block diagram
4.1 Component selection

The first step in the design process is to find the components best suited for this project. Choices are based on performance, cost and availability. Each of the major blocks will be discussed separately.

4.1.1 RF link

The first block to consider is that of the RF link. In order to design this block, certain design specifications have to be stated. The frequency band that must be used is that of the 433 MHz ISM band, while the required transmission radius must be approximately 1 km. According to [Tec06] a distance of between 1-2 km can be obtained with 10 dBm transmit power at 433 MHz if a line-of-sight (LOS) link is available. With sufficiently high antenna masts almost all of the stations would have a LOS link, as explained in Chapter 2. A minimum transmit power of 10 dBm is therefore required. Products from the following manufacturers were considered:

- **Analog devices** - (ADF7020, ADF7021 and ADF7025)
  
  All three of these devices support the frequency band required. The data rates vary from 0.3 kbps to 312 kbps, with an operating voltage range between 2.3 V and 3.6 V. All three of these chips support a programmable output power between -16 dBm and 13 dBm. Their receiver sensitivity varies between -101 dBm and -117 dBm, depending on the modulation method used. The ADF7025 chip has an added on-chip VCO and functional-N PLL. The difference between the ADF7020 and the ADF7021 is that the ADF7021 uses a narrower band. These chips interface with a microcontroller via a SPI interface and 4 other control signals. With these chips all forms of communication management such as preambles, postambles, FEC codes, etc. have to be managed manually.

- **Microchip (rfPIC12F675F)**
  
  This chip has its own microcontroller with 1 kB flash program memory and 128 bytes EEPROM. The output power can be set between -12 dBm and 10 dBm while using ASK
or FSK modulation. The transmit frequency can be set between 380 MHz and 450 MHz, with a maximum data rate of 40 kbps. The chip, however, has only a transmitter and a separate receiver would have to be used. The microcontroller is not that powerful and the number of ADC ports is limited. An additional microcontroller would, therefore, have to be used.

- **Nordic Semiconductor (nRF905 and nRF9E5)**

  The nRF905 chip uses GFSK modulation with Manchester encoding, delivering a maximum bit rate of 50 kbps. GFSK modulation results in a more bandwidth effective transmission-link compared with ordinary FSK modulation. This chip also supports multi-channel communication and is completely ETSI and FCC compatible in the 433 MHz, 868 MHz and 915 MHz ISM bands. The output power can be varied between -10 dBm and 10 dBm, while the receiver has a sensitivity of approximately -100 dBm. The operating voltage of this chip is between 1.9 V and 3.6 V. This chip supports automatic preamble generation and CRC padding and automatically checks whether the received CRC check is valid. The nRF905 transceiver also supports carrier detection and address matching and uses an external data ready pin to indicate when valid data has been transmitted and received. This chip interfaces with a microcontroller via a SPI interface and 6 additional control pins. Another advantage of this chip is its extremely low cost (also referred to as Bill of materials) and low transition times.

  The nRF905 transceiver is the most obvious choice. The simple interface used and the way this chip handles data, makes it very easy to manage the RF link. The nRF9E5 microcontroller is not used because this chip does not have on-chip program memory, which forces the use of an external SPI EEPROM. The nRF9E5 also does not have enough ADC ports and it only supports the use of a software implemented RTC. The only prerequisite left to satisfy, is that of the transmission range. According to [SKHH06] transmission ranges up to 1.1 km have been measured in an implemented sensor network which used the nRF905 chip together with antennas with gains of roughly 2 dBi. The link quality was also simulated with Radio Mobile, which is a program that simulates and predicts signal propagation. This program can be downloaded from http://www.cplus.org/rmw/english1.html. The results obtained are shown in Fig. 4.3. Note that the received power over 1.15 km is 7.1 dB with the antenna gain set to 0 dBi. The nRF905 chip therefore satisfies all the prerequisites. The connection of this chip, together with all the external components needed, is laid out in the device datasheet.
4.1.2 Ethernet interface

This link does not have to be that fast, because the RF links used have a maximum data rate of 50 kbps. The easiest and cheapest solution should be used. Chips from two different vendors were considered. They are:

- **Silicon Laboratories (CP2200)**
  This chip has a fully integrated IEEE 802.3 MAC and 10 BASE-T PHY and is fully compatible with 100/1000 BASE-T networks. The CP2200 supports both duplex and half duplex communication and is available in both a multiplexed QFN-28 pin package and a non-multiplexed 48-pin TQFP package. This chip can be controlled by a microcontroller via an 8-bit parallel interface with a minimum of three additional control I/O pins. This chip supports automatic padding and CRC generation as well as automatic retransmission on collision. This chip has 8kB of on-chip flash memory with a 2 kB RAM TX buffer and a 4kB RAM RX buffer. No external memory is therefore required. The operating voltage of this chip is 3.1 V to 3.6 V. Source code is freely available for the TCP/IP stack and device drivers.

- **Microchip (ENC28J60 and PIC18F67J60)**
  The ENC28J60 is available in a 28-Pin SPDIP, SSOP, SOIC and QFN package. It has all the basic features supplied by the CP2200 controller. A microcontroller can control this chip via a SPI interface and 2 additional control lines. The chip has 8 kB transmit/receive packet dual port SRAM and also performs hardware assisted IP checksum calculations. The operating voltage of this chip is between 3.14 V and 3.45 V.
The PIC18F67J60 chip is a microcontroller with an embedded ENC28J60 ethernet controller. It has 128 kB flash program memory and 3 kB SRAM data memory. This chip also has a SPI port, ADC ports (10-bit) and multiple timers. This chip is perfect as a microcontroller and ethernet interface, but was not yet in production at the time this project was initiated. However, all datasheets were available at that time.

The ENC28J60 ethernet controller was chosen as ethernet interface for the APB because of its relatively easy interface and it is also the cheapest of the options considered. An example circuit for this chip is shown in the device datasheet.

### 4.1.3 Microcontroller

The components used thus far for both the RF link and the ethernet link are interfaced via a SPI interface. If a microcontroller with only one SPI port is used, additional circuitry would be needed to implement some sort of multiplexing. The easiest solution would therefore be to use a controller with two SPI ports. The ADC interface can be obtained either by using an external ADC port, or by using a microcontroller with on-chip ADC ports. At least 5 ADC ports are needed. The board designed must be able to connect to a PC via a serial connection. The controller used must, therefore, contain a RS-232 port. Another important requirement is that of both program memory and data memory. It would simplify the design enormously if no external memory were required. It should be remembered that sufficient memory (EEPROM) should also be available to store parameters such as station names, settings, etc. Both the RF link and the ethernet link require two interrupt pins to control data flow. Four external interrupt pins are therefore required. Time stamping of data is also needed, which requires the use of a RTC module. This can be implemented externally or internally. The chips considered were narrowed down to the three options shown in Table 4.1.

<table>
<thead>
<tr>
<th>Device</th>
<th>Package</th>
<th>Program Memory</th>
<th>EEPROM</th>
<th>RAM</th>
<th>10-bit ADC</th>
<th>External Interrupts</th>
<th>RTC</th>
<th>Timers</th>
<th>WDT</th>
<th>Operating range</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC18F6722</td>
<td>64 TQFP</td>
<td>128kB</td>
<td>1kB</td>
<td>3.8kB</td>
<td>12</td>
<td>4</td>
<td>Software</td>
<td>3-16 bit, 2 8 bit</td>
<td>yes</td>
<td>2.0V-5.5V</td>
</tr>
<tr>
<td>dsPIC30F6015</td>
<td>64 TQFP</td>
<td>144kB</td>
<td>4kB</td>
<td>8kB</td>
<td>16</td>
<td>5</td>
<td>Software</td>
<td>5-16 bit</td>
<td>yes</td>
<td>2.5V-5.5V</td>
</tr>
<tr>
<td>PIC24FJ128GA006</td>
<td>64 TGFP</td>
<td>128kB</td>
<td>-</td>
<td>8kB</td>
<td>16</td>
<td>5</td>
<td>Hardware (RTCC)</td>
<td>5-16 bit</td>
<td>yes</td>
<td>2.0V-3.6V</td>
</tr>
</tbody>
</table>

**Table 4.1: Microcontrollers considered**

All three of these chips have two SPI ports and two UART ports. The PIC18F is an 8 bit controller, while the dsPIC30F and the PIC24F are 16 bit controllers. Both the RF link and the ethernet link chosen thus far can operate at 3.3 V, which is within the voltage range of all three of these controllers. This reduces the design requirements considerably, because only one voltage rail is needed and no additional circuitry is required between the chips for communication. Note that the PIC24F does not have on-chip EEPROM. This shortage is overcome by the introduction
of run-time self-programming (RTSP), which allows the controller to manually write code to its own program memory. A small percentage of the PIC24F’s program memory can therefore be used to create a virtual EEPROM. It should be stated that the number of writes allowed to a sector of memory is approximately 1000. This is a valid replacement, because the memory will only have to be altered if the settings change, which would not happen very often. Techniques also exist to test whether a write was successful or not. If not, the data is written to a different block which has not yet been written to. By implementing such schemes, the number of write cycles can be increased if necessary. The PIC24F is the only controller of the three options that has a hardware RTCC. The PIC24FJ128GA006 microcontroller was chosen.

4.1.4 Serial interface

All components chosen thus far can operate at 3.3 V and, therefore, this voltage has been chosen as a voltage rail. The MAX3232 serial transceiver manufactured by MAXIM is chosen to implement the serial interface, as it also operates at 3.3 V. Additional capacitors are used together with this chip to build charge pumps to allow communication between this chip and a PC, which uses much higher voltages for communication. This chip can withstand input voltages of up to 25 V on its receive pins and no buffer is required between a PC and this chip. An example of how this chip should be connected can be found in the device datasheet.

4.1.5 Power supply

All the sites that have to be monitored by this network are fed by AC power, which greatly simplifies the power supply module. A standard AC to DC adapter is used to convert the AC voltage to a DC voltage between 6 V and 9 V. To ensure a stable voltage with minimal fluctuations, a 3.3 V low drop out voltage regulator is used to regulate the supplied voltage. The absolute maximum current that can be drawn by the PIC24F is 300 mA, by the ENC28J60 250 mA and by the nRF905 28 mA. The maximum current that can be drawn by this board is roughly 600 mA. The LM1117 voltage regulator manufactured by National Semiconductor can supply 3.3 V at a maximum current consumption of 800 mA. The actual current consumption of this circuit is much lower than this maximum value. Refer to Section 8 for the actual measured results.

4.2 Circuit layout

4.2.1 Connection of the nRF905 RF transceiver

The complete layout of the nRF905 chip with its accompanying circuitry is shown in the nRF905 datasheet. It is very important that this chip must be thoroughly grounded. A ground plane
should be added directly below the chip and should also be connected to the bottom layer using a through-hole via. The layout of the matching network should be exactly the same as that shown in the datasheet. Small changes in track length, track width or component clearance can severely affect the matching RF circuit. Even the use of different components (footprint or manufacturer) has an enormous influence on the matching network. The circuit designs described in this thesis used components with 0805 footprints and not 0603 footprints as used in the datasheet design. This resulted in an RF loss of approximately 20 dB. For testing purposes this was perfect, because of the limited space available for testing (see Section 8.2). The complete matching circuit with the correct grounding, matching and external components can be bought as a complete packet, which is interfaced via a dip-24 footprint. This complete circuit is manufactured by Polygon Technologies and is shown in Fig. 4.4. The RF power transmitted by this circuit was tested and is shown in Section 8.1.

With both the initial RSB and APB, the 4 SPI lines are connected to the SPI 2 port of the microcontroller. Note that the MISO and MOSI lines are connected the wrong way around in the initial designs (Fig. 4.10 and 4.11), but correctly in the final RSB design (Fig. 4.12). The transceiver is controlled via six pins. They are DR, AM, CD, PWR_UP, TXEN and TRX_CE. The first three pins are input pins and the microcontroller uses them to detect RF data status. When data is transmitted or received correctly, the nRF905 transceiver sets the DR (data ready) pin high, while the AM (address match) pin is used to indicate that a correct destination address has been received. The DR and AM pins indicate when data is received or transmitted correctly and are connected to the external interrupt pins INT3 and INT4 respectively. This allows the RF link to be interrupt driven. The CD (carrier detect) pin is set high when a carrier is detected and is used for channel sniffing. The other three pins are output pins and are used to control the state of the transceiver. The PWR_UP pin is used to power up the nRF905 chip, while the TRX_CE pin is used to enable the transceiver (transmit or receive). The TXEN pin is used to indicate whether data should be transmitted or received if the TRX_CE pin is set. These three pins, together with the CD pin, are connected to GPIO pins.

In the final RSB design (shown in Fig. 4.12) a second transceiver was added to test the effect of dual communication channels. The same procedure as explained above was followed to connect the second transceiver. The 4 SPI lines were connected to the SPI 1 port and the DR and AM
pins connected to INT1 and INT2 respectively. The other 4 control lines were connected to GPIO pins.

The 50 ohm matched RF output of the nRF905 transceiver is connected to a standard SMA RF connector. Any type of antenna matched to 50 ohm can therefore be connected to the RSB. For testing purposes, a 1/4 wave whip antenna was used which simply comprises a 1/4 wavelength conductor mounted directly onto the RSB. This type of antenna is ground-dependent and requires an external reference plane to radiate power efficiently. When an infinitely large ground plane is used, the whip antenna performs exactly the same as a dipole antenna (\cite{Poz01}). A whip antenna has an omni-directional radiation pattern and is, therefore, ideal for this type of network. The length of the antenna can be determined as 17.3 cm using eq. 4.1. To make this antenna, a piece of conductive wire is cut 17.3 cm long and one end connected to the RF output. This antenna is heavily mismatched and the ground plane used is also very small. Therefore the amount of radiated power transmitted and received decreases enormously. For testing purposes this is perfect, as explained in Section 8.2. Typical properties of antennas that can be used at this frequency are shown in Table 4.2 (table taken from \cite{Tec06}).

\[
L = \frac{\lambda}{4} = \frac{69.3}{4} = 17.3\,\text{cm}; \quad \text{with} \quad \lambda = \frac{c_0}{f} = \frac{3 \times 10^8}{433 \times 10^6} = 69.3\,\text{cm} \quad (4.1)
\]

\[4.2.2 \text{ Connection of the ENC28J60 ethernet controller}\]

This interface was designed and built, but was not implemented. The four SPI lines are connected to the SPI 1 port of the microcontroller. The ENC28J60 controller has seven interrupt sources, but only two interrupt pins, \text{INT} and \text{WOL}. The \text{INT} interrupt represents all control events, while the \text{WOL} interrupt represents all the wake-ups on LAN events. These two pins are connected to the INT1 and INT2 external interrupts of the microcontroller respectively. This allows the ethernet port to be controlled by an interrupt driven software driver. The only other pin connected to the controller is that of a \text{RESET} pin, which can be used to reset the ENC28J60 device via software. Several external components are required to complete the ethernet interface. The most important component is that of the decoupling transformers, which should be rated for isolation of at least 2 kV. This isolation is necessary to protect circuits connected to the ethernet from static voltages. The RJ-45 ethernet connector used has embedded magnetics which performs the isolation and no extra transformers are therefore needed. The complete

<table>
<thead>
<tr>
<th>Antenna Type</th>
<th>Radiation Pattern</th>
<th>Typical Gain (dBi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4-Wave Whip</td>
<td>Omni</td>
<td>-3 to 0</td>
</tr>
<tr>
<td>Dipole</td>
<td>Omni</td>
<td>2</td>
</tr>
<tr>
<td>Collinear</td>
<td>Omni</td>
<td>5, 8 or 10</td>
</tr>
<tr>
<td>Yagi-2 to 16 elements</td>
<td>Directional</td>
<td>4 to 16</td>
</tr>
</tbody>
</table>
design of the ethernet circuitry can be found in the ENC28J60 datasheet.

### 4.2.3 Connection of the MAX3232 serial transceiver

The complete connection of the serial transceiver is given in the device datasheet. This transceiver is connected to the UART 2 port of the microcontroller. Even though no form of hardware control is used for communication, the RTS and CTS pins are still connected.

### 4.2.4 Connection of the ADC port

All the ADC ports available after all the other peripherals have been connected, are used. They are routed to one single connection point on the board which could be accessed via a dual header connector. The number of ADC ports available with the initial RSB circuit is 10, while 9 ports are available with the final RSB circuit. No external ADC circuitry was added onto the board, due to the numerous different types of transducer interfaces available. One of the most commonly used interfaces uses varying currents, between 4 mA and 20 mA, to indicate different levels. Different voltage interfaces also exist, but are very difficult to design for because they vary enormously depending on the transducer used (For example 25 mV-38 mV and 0-10 V).

The reference for the ADC port of the microcontroller can be set via the $V_{\text{ref}+}$ and $V_{\text{ref}-}$ pins or it can be chosen as $V_{\text{SS}}$ and $V_{\text{DD}}$ in the configuration registers. The latter is implemented and, therefore, all interfaces should be between 0 V and 3.3 V. A typical circuit used to convert the 4-20 mA interface to the 0-3.3 V interface is shown in Fig. 4.5. The way the operational amplifier is connected is referred to as a voltage follower configuration. An operational amplifier has a very high input impedance and a low output source impedance and therefore serves as a buffer between the transducer and the microcontroller. The input current is converted to a voltage by sending the current through a grounded resistor. The maximum output voltage allowed by this circuit is 3.3 V, which must represent a 20 mA input current. The resistor value $R_{\text{in}}$ can be determined by eq. 4.2 as $165\,\Omega$. The minimum value that can be obtained is therefore $0.66\,\text{V} ((4 \times 10^{-3}) (165))$. Note that a negative supply ($V_{\text{–}}$) must be applied to the operational amplifier if the amplifier cannot swing as low as 0.66 V. The minimum swing possible is amplifier dependent. It is very important that the ground reference of the transducer and the RSB should be connected.

\[ R_{\text{in}} = \frac{V_{\text{out}}}{i_{\text{in}}} = \frac{3.3}{20 \times 10^{-3}} = 165\,\Omega \quad (4.2) \]

### 4.2.5 Connecting the power supply

The DC power is connected to the RSB via a DC socket and a diode is added in series to protect the circuit, in case of the power being connected the wrong way around. The LM1117 voltage regulator is connected in the fixed output topology shown in its datasheet. The two
Tantalum capacitors are used to keep the voltage rails stable. Almost all components generate some form of oscillation or noise. Even though this might be very small, they interfere with each other and could cause the entire circuit to oscillate or malfunction. Typical sources of these oscillations are the crystals of each of the controllers, the RF transmitter, small AC ripples from the power supply, the SPI clocks etc. To reduce the interference caused by these oscillations, all the different modules are kept separate. Their ground planes are connected, but the $V_{DD}$ source is decoupled using RF chokes. These chokes have low impedance at low frequencies, but high impedance at higher frequencies. These chokes therefore do not influence the DC power, but filter all the high frequency oscillations. No decoupling was used with the initial boards, but it was implemented in the final design (See L7, L8 and L9 in Fig. 4.12).

### 4.2.6 External connections of the microcontroller

The core digital logic of the PIC24FJ128GA must be supplied with a voltage of 2.5 V. This chip has an on-chip LDO regulator which can be used to supply this voltage. This internal LDO regulator takes its input voltage from the external voltage ($V_{DD}$). To enable this regulator, the ENVREG pin must be connected to the $V_{DD}$ pin and an external 10uF tantalum capacitor must be connected between the $V_{DDCORE}/V_{CAP}$ pin and ground (See Fig. 4.7, taken from the device datasheet). The tantalum capacitor has a low effective series resistance (ESR) and helps to stabilize the regulator.

Even though the controller has a internal fast RC oscillator (FRC) of 8 MHz, an external 8 MHz oscillator is nevertheless used. According to [Mic06] the frequency obtained using the 4xPLL with the FRC oscillator tends to be less precise than that obtained using the external primary oscillator. The controller requires the use of parallel cut crystals and additional capacitors to supply a stable frequency. According to [Mic06], capacitors $C_1$ and $C_2$ shown in Fig. 4.6 should be 22pF. An additional 1MΩ resistor is added in parallel with the crystal to improve stability.

The microcontroller is programmed with a MPLAB ICD2 programmer via an in circuit serial
programmer (ICSP) interface. The ICD2 programmer connects to the RSB via a RJ11 connector, shown in Fig. 4.9. As shown in Fig. 4.8, the programmer uses three lines, those of PGC, PGD and MCLR, to program the microcontroller. The PGC line is used to send clock pulses, while the PGD line is used to send data. The 10kΩ pull up resistor on the MCLR line is very important, because it pulls that line high if the programmer is not connected and, therefore, prevents the memory from being cleared. A tactile switch is added to the circuit to replace the user reset block and can be used as a reset switch.

The final addition to the RSB is that of decoupling capacitors and LED indicators. Decoupling capacitors were added to all VDD pins. A decoupling capacitor is used to decouple one part of a network from another, by shunting any noise caused by circuit elements via this capacitor to ground. A total of 8 LEDs were added to use as indicators. The current through each resistor was limited to a maximum of 4 mA to reduce the total power dissipated by the microcontroller. The final RSB circuit designed is shown in Fig. 4.12.

4.3 PCB design and manufacturing

The PCBs were designed using the Protel software. The initial RSB and APB boards were manufactured at the engineering faculty of the University of Stellenbosch. These boards were
Figure 4.10: The initial RSB layout

only used as prototypes. The designed track widths were too narrow to manufacture in-house and were adjusted by the manufacturer. The manufacturing capabilities also did not support silk screens, solder masks or through-hole plating. All the vias had to be connected manually and the tracks were very fragile. The nRF905 chip’s grounding was also not very good. The final RSB circuit with the dual RF channels was manufactured by North Tech Services. The top layer, bottom layer and silkscreen of this design are shown in Fig. 4.13, 4.14 and 4.15 respectively. These circuits were through-hole plated and a solder mask was also added. A picture of the final product is shown in Fig. 4.3. Note the two 1/4 wave whip antennas used. The antennas which were used, are a very low cost solution. The antenna properties of these antennas are not very good and a better antenna should be used if long distance transmission is required.

4.4 Cost

An important aim of this project is to minimize cost, while retaining functionality. The cost of manufacturing one of the final RSB stations is approximately R 330. This price excludes the initial cost of approximately R 550 charged by the PCB manufacturer to set up the project.
Figure 4.11: The initial APB layout
**Figure 4.12:** The final RSB layout
Figure 4.13: PCB Top Layer (Scale 1:1)

Figure 4.14: PCB Silkscreen (Scale 1:1)
The cost of this initial setup depends on the circuit size and layout. This setup cost is paid only once and thereafter each station costs R 72, which is included in the RSB cost calculation. The second RF transceiver was only added to test the performance of a dual channel system. Considering the traffic density in this project the second RF transceiver could be removed, which would decrease the price by approximately R 50. When using the completed RF modules shown in Fig. 4.4, the price increases to approximately R 480.

To increase the transmission range of this circuit, a higher gain antenna should be used to increase the link quality. A typical choice would be that of a 2 dBi dipole or collinear antenna. The cost of these antennas varies between R 120 and R 600. Also note that the estimated cost does not include any transducers.
Figure 4.16: Photo of the RSB

Table 4.3: Total cost of one RSB PCB
Chapter 5

Embedded PIC software

Any device that includes a programmable computer, but is not intended to function as a general-purpose computer itself, is called an embedded computer system ([Wol01]). The hardware designed in Chapter 4 is an example of such a system. This hardware cannot function on its own and requires some embedded software to control it. In this chapter the design of the embedded program will be discussed.

5.1 Basic design

The embedded software was written in C using MPLAB IDE (Integrated Development Environment). MPLAB IDE is only a GUI front end and an additional compiler is required to compile the code. The PIC24F is a 16 bit microcontroller which requires the use of the MPLAB C30 compiler. The C30 compiler optimizes the C code and compiles it into assembly language files. These files are then linked with other object files and libraries to produce an executable COFF or ELF file which is sent to MPLAB IDE where this file can be converted to HEX format. This HEX file can then be loaded onto the PIC via a programmer such as the MPLAB ICD. The tasks performed by the C30 compiler are shown in Fig. 5.1, which was taken from [MPL05]. Both these development tools are freely available from the Microchip website. The PIC24F is a relatively new controller and standard C30 libraries are not yet available for all the modules, but numerous code examples can be downloaded from the Microchip website.

5.2 Controller configuration

Before any code can be executed, the configuration settings of the PIC must be set. These settings include oscillator configurations, watchdog timer settings and ICSP port selection. For the RSB, the primary oscillator is used in HS mode with clock switching disabled. These settings provide a 32 MHz clock to the controller. The watchdog timer is also enabled with a period
of approximately 2 seconds. The secondary oscillator is also used as source for the RTCC, but this is not set in the configuration settings. For the enabling procedure of this oscillator, refer to Section 5.4.3.

### 5.3 Main program

The main PIC program must control all the interfaces and the data management. This program can be set up to poll all the different procedures continuously to check whether they have data to be processed or not. This method could cause some procedures to wait quite a long time before being serviced, especially if the procedures are very long. To overcome this problem, all the different peripherals are written to be interrupt driven. The embedded program can, therefore, be subdivided into module blocks which are interrupt driven (see Fig. 5.3). The flowchart of the main program is shown in Fig. 5.2. As seen in this figure, the main program is only used to initialize all the interfaces and procedures. Note the RTSP test block in particular. This block tests whether the RSB settings have previously been stored in the program memory using RTSP (refer to Section 5.5.5). If settings are detected in the program memory, all the blocks are initialized using those values, otherwise they are initialized according to the manufacturer’s default values. After initializing all the blocks and enabling all the interrupts, the program must continue in a endless loop. This loop consists of clearing the watchdog timer and scanning the RSB’s status. The use of the WDT is explained in Section 5.4.7. The status of the RSB is continuously scanned and indicated using LEDs. The states monitored according to LED indicators are:

- **RS232** - This LED indicates whether the RSB is connected to a PC or not.
- **ACK** - This LED indicates that a message has been sent and the RSB is still waiting for
the ACK.

- **Level0** - This LED indicates that the network has not yet detected this RSB and it is currently not connected to the network (Level 0).

- **ScanREQ** - Used to indicate that this station has received a scan request signal from a lower level station.

- **TestLED** - Used for testing purposes (Debugging).

- **Scan** - Used to indicate that this station is currently scanning for children.

- **Detect** - Used to indicate that a station is busy with a grandparent scan. Refer to Chapter 3 for more detail on the different types of scanning procedures.

![Flowchart of the main PIC program](image)

**Figure 5.2:** Flowchart of the main PIC program

### 5.4 Interrupt service routines

The 7 interrupt blocks mentioned earlier are shown in Fig. 5.3. Note the arrows indicating the direction in which these interrupts influence the other blocks. Each of the 7 main blocks generates an interrupt in the main program. Each interrupt has a priority assigned to it, the highest priority being 7. An interrupt can interrupt another if its priority is higher than that of the other interrupt. To prevent this from happening, a software interrupt handler is used
to disable any interrupts that can cause problems. After servicing an interrupt, the handler re-enables the interrupts. If an event has occurred during the time the interrupt was disabled, it would be generated immediately after being re-enabled. This happens because the interrupt flag for that specific routine is set. It should always be remembered that an interrupt must be disabled while servicing it and the interrupt flag be cleared manually. The seven interrupt blocks will now be discussed individually.

![Diagram](image)

**Figure 5.3: Different interrupts**

### 5.4.1 RF interrupt

A RF interrupt is triggered by the DR pins of both the RF1 and RF2 controllers connected to the INT1 and INT3 external interrupt pins respectively. The different interrupts are referred to as DR1 and DR2 respectively, or DRx if referred to any one of these interrupts. A DR (data ready) interrupt occurs whenever data has been sent or received successfully. For valid data to be received a valid address must already have been received, which would set the AM (address match) pin. The AM pin would not be set if data was sent. The AM pin is therefore used to determine if a DR interrupt was caused after transmission or reception of data. The flowchart for the DR interrupt caused by one of the transceivers is shown in Fig. 5.4. Note that both DR interrupts are disabled when a DR interrupt was caused after transmission or reception of data. The figure clearly shows that the transmitter is set as available, and RX mode, if the DR interrupt occurred due to successfully transmitting data. The RF interrupt block consists of the DR interrupts generated by both RF transceivers. RF transmission is the most important aspect of the RSB and therefore its interrupt has priority over all the other interrupts. The priority is set to 6.

The complete RF command set implemented can be found in Section 3.5.1. The communication strategy implemented is thoroughly discussed in Chapter 3. The state of each RF transceiver must be known at all times. Each RF transceiver therefore has a status register containing all the required information. These states are stored in a structured union called `nRFxflags`. The
Chapter 5 — Embedded PIC software

Figure 5.4: Flowchart of the DR interrupt

members of this structure and their corresponding tasks are:

• LAST_SPI - This variable contains a time stamp of when data was last sent to the RX register to be transmitted. This time stamp is used to determine when the TX entry can be cleared (buffer available), if the data has not been transmitted after a certain time (TTL).

• RX - This 1 bit flag is used to indicate that data has been received and must be serviced. After this data has been serviced, this flag is cleared.

• TX - This 1 bit flag indicates that the TX buffer is currently occupied. It is cleared in the DR interrupt procedure after the data has been transmitted, or when the LAST_SPI TTL expires.

• TX_EN - 1 bit indicating whether the TX_EN pin is set or not.

• TRX_CE - 1 bit indicating that the TRX_CE pin is set.

• PWR_UP - 1 bit indicating whether the nRF905 chip is powered up.

• CARRIER - Set if the CD pins detect a carrier on this channel.

• BACKOFF - This 1 bit flag shows that the current message in the TX buffer is in back-off mode.
5.4.2 Serial interrupt

This interrupt is triggered only when data is received on its serial port. The serial interface is set to run at 115k baud with 8 bits data, no parity and 1 stop bit. An end sequence is used to indicate the end of the serial data. This sequence consists of the line feed (decimal 10) and carriage return (decimal 13) characters. When data is sent to the PIC, \#10\#13 is used, but this sequence is repeated twice (\#10\#13\#10\#13) when data is sent to the PC. Data is read from the UART RX buffer until this end sequence is found. It should be stated that the indexing variable, Data\_RX\_pos is initialized to 0 in the “initialize variables and states” block shown in Fig. 5.2. Data sent via the serial port is limited to 40 characters including the end sequence and command byte. After all the data has been read from the buffer, it is processed by the UART\_RX\_Choice procedure. The first character contains the command. Refer to Table 6.1 in Section 6.2.1 for the complete serial command set. The flowchart of the serial interrupt is shown in Fig. 5.5.

![Flowchart of the serial interrupt](image)

Figure 5.5: Flowchart of the serial interrupt

5.4.3 Real-Time Clock/Calendar interrupt (RTCC)

A 32.768 kHz crystal is connected as secondary oscillator (SOSC), which is used as clock source for the hardware RTCC. This clock is used in conjunction with the RTCC prescaler to generate a half second clock. This clock cycle time is not always exactly 0.5 s, depending on the oscillator used, but it can be calibrated using the RCFGCAL register. The SOSC is enabled by setting the SOSCEN bit in the oscillator configuration register (OSCCON). An unlock sequence is required before a write to this register can occur to prevent accidental changes in the oscillator configuration.

The RTCC interrupt is triggered by the alarm function. The time between consecutive alarms
can be set via the GUI. The available intervals range between half a second to one year. The default interval is 1 second. As seen in Fig. 5.3 this interrupt service routine is responsible for network rescans, drop of lost stations, monitoring of ADC ports and to stop failed GP scans. A variable is set via the GUI which determines how many interrupts must occur before a full network scan is initiated. Another variable is set to determine how many interrupts should occur before a station is dropped from the network. This only happens when a scan sequence was not detected during the previous scan. If a scan packet is received, this drop period is reset to 0. The drop period must always be higher than that of the scan period, to prevent all stations from being lost while waiting for the next scan sequence. This service routine also detects when a GP scan error occurs. This happens when a scan request signal is successfully sent to a station and the ACK is received back. The station scanning the network would then send a SetLVL command. If the valid ACK for this message is not received, the GP scan procedure will get stuck, because the scanning station has already detected the station, but has not received confirmation. For more detail on functioning of the scanning procedure, refer to Chapter 3. The last task of this routine is to scan the ADC ports. This happens every time the interrupt occurs.

Every ADC set to be monitored will be scanned eight times and the average taken. If the measured value is within 10% of the previous measured value, it would be dropped, otherwise the new value will be sent to the server and the previous value will also be adjusted. Only values changing by more than 10% are sent, to reduce the total traffic density. Each ADC port has its own structure containing all the information required on the transducer connected to that port, and its status. A total of 16 ADC ports are present for this station and, therefore, an array containing 16 of these structures is used for each RSB. This complete structure is stored in the program memory using RTSP, if the change memory setting is used. This provides knowledge of which ports to monitor, even after a power failure. The fields of this structure and their functions are:

- **INFO** - This single byte is used to store the type of transducer connected to the port. All the different types of transducer are given a code at the GUI which coincides with this value. A total of 255 different types of transducer can therefore be used. Note that these values are not RSB specific, but transducer type specific (e.g. 4-20 mA pressure sensor, 0-10 V depth sensor, etc.).

- **VALUE** - An integer containing the previous value measured. Note that this value is not updated if a new measured value is within 10% of this value.

- **USE** - A single flag bit used to indicate whether this port should be monitored or not.

- **CHANGED** - A single flag bit used to indicate weather this ADC port’s value changed during this scan. This flag is cleared once the new data is sent to the server.
5.4.4 Back-off interrupt

This interrupt is triggered by the timer 2 (T2) and timer 3 (T3) interrupts. These two timers are used for packets sent via channel 1 and channel 2 respectively. Both T2 and T3 are set to interrupt every millisecond, but are only activated when a message is sent into back-off mode. When a message is sent into back-off, a random number between 1 and a value set in the GUI is chosen and assigned to that packet. This number represents the time (in ms) that this packet should spend in back-off mode before retransmission. Note that T2 and T3 work completely independently. The flowchart for this interrupt is shown in Fig. 5.7.
5.4.5 Service routine interrupt

This interrupt is triggered by the timer 4 interrupt with inter arrival times set via the GUI (default time is 40 ms). This ISR (interrupt service routine) is used to scan the ACK stack for expired TTL timers and to release the TX buffer if it has been occupied too long without ever transmitting data. The flowchart for this routine is shown in Fig. 5.8. Note how all the interrupts are disabled in this ISR to prevent any data flow from occurring. The reason for this is that scanning the ACK stack can generate new data transmission and also start new scan procedures. Therefore any clashes should be prevented. This service routine is used for TTL scanning, rather than putting it in the main loop, because all interrupts must be disabled. If this code was added to the main loop, it would not have been possible for the system to be interrupt driven.

5.4.6 Scan button

This interrupt is generated when a change in pin CN13 occurs (CNinterrupt). This interrupt can either set the RSB as base station, if it is currently not connected to the network (level is 0), or it starts a new scan sequence. This is used to manually set up the network without the GUI. A reset button can be connected to pin CN13 if this feature is desired. The flow chart of this interrupt is shown in Fig. 5.9.

5.4.7 Watchdog timer interrupt (WDT)

The WDT's clock source is the internal low power RC oscillator (LPRC). This timer can be varied between 1ms and 131 s using post scalers. In this project, this interrupt is set up to be triggered approximately every 2 seconds. When this interrupt occurs, it resets the PIC. To

![Flowchart of the back-off interrupt](image-url)
Figure 5.8: Flowchart of the service routine interrupt

prevent this reset, the WDT has to be cleared continuously to prevent the timer from overflowing. This is done in the main program loop shown in Fig. 5.2.

Figure 5.9: Flowchart of the push button interrupt
5.5 Other software techniques implemented

5.5.1 Delay (Timer1)

The RF transceiver requires some transition times when switching between states. To ensure proper functionality, delays must be implemented to account for these switching times. The delays required vary between 550 us and 3 ms and are implemented using timer 1 and a counter. Timer 1’s period is set to 1 us, while the counter is decremented every time timer 1 expires. The data type of the counter is an integer, which allows the delay to be anything between 1 us and 32767 us. This timer is not interrupt driven and its flowchart is shown in Fig. 5.10.

![Flowchart of the delay implemented using timer 1](image)

Figure 5.10: Flowchart of the delay implemented using timer 1

5.5.2 Random values

The back-off times should be completely random. The \texttt{rand} function generates pseudo random values according to a predefined seed. A pseudo random number generator (PRNG) uses an algorithm which generates values with the same statistical properties as a random process, but has some element of predictability. To ensure that the random values generated are different for each RSB, the seed for each station is set individually during initialization. The seed is set with the \texttt{srand} function. The algorithm used to determine a station’s seed uses a station key to determine the seed. This key is derived by adding the ASCII values of the first and the third character of the station ID and then subtracting the second and fourth character. This value is used as the seed to generate two random numbers. The absolute value of the difference between these two random numbers is then used as the station’s seed.

5.5.3 Timestamp

Every packet that is transmitted must be tagged with a timestamp when sent to the ACK stack. This timestamp is used to determine TTL expirations. The RTCC module was initially used for time stamping, but was replaced with a timer to reduce processing time and increase accuracy. When using the RTCC, the complete register has to be read and converted every
time a timestamp is needed. Another problem with the RTCC is that the time resolution is 0.5 s, while typical TTL settings is 40-100 ms. Timer 5 is a 16 bit timer and is set to overflow every 2 s. Each increment in the timer therefore represents a 30.52 us \(\left(\frac{4}{2^{16}}\right)\) time delay. The TTL tags are in ms and therefore, a scaling factor should be used to convert each step into ms. This factor is 32.77 \(\left(\frac{1000\text{us}}{30.52\text{us}}\right)\). Whenever a timestamp is needed, the value in the TMR5 register should be taken and divided by 32.77, which would supply the timestamp in ms. The time in the ACK stack can easily be determined by subtracting the packet’s timestamp from the current timestamp. If the current timestamp is less than a packet time stamp, an overflow occurred and the subtraction terms should be swapped around. Note that this maximum timer range of 2 s limits the maximum allowable TTL. If the TTL for one hop is 100 ms, a maximum of 20 hops will be allowed before overflow errors would occur.

### 5.5.4 SPI module

Data communication between the RF transceiver and the microcontroller is obtained via a SPI interface. The nRF905 controller supports SPI mode 0, which implies that data is latched at the rising edge of theSPI clock. The PIC is set up as master and the nRF905 chips as slaves. Non-framed byte wide communication is used with a clock speed of 4 MHz.

### 5.5.5 Run-Time Self-Programming (RTSP)

The PIC24FJ128GA006 microcontroller does not have on-chip EEPROM. To reduce the cost of adding an external EEPROM, the process of RTSP is introduced. With RTSP, data is written to the program memory by the controller itself during normal operation, using the table read (TBLRD) and table write (TBLWT) instructions. The PIC24F controller’s program memory is addressed as shown in Fig. 5.11. Note that the upper byte of the high word is not implemented and should not be altered. When using RTSP proper planning is a necessity to ensure that program code is not overwritten, deleted or moved. The PIC24 organizes its flash memory into rows containing 64 instructions (192 bytes). Data can be written 1 row (192 bytes) at a time and can be erased 8 rows (1536 bytes) at a time. The basic rule of thumb with RTSP, is that any given word in memory must not be written more than twice before erasing the page in which it is located. With this rule kept in mind, the procedure used when updating the memory block, is to first read the entire page into a variable. The new values are then updated in this variable before the entire page is erased. After erasing the block the new updated variable is written to the program memory. To prevent accidental memory operations, RTSP requires an unlock sequence, before a programming sequence can be initiated. A complete RTSP cycle typically takes about 4 ms during which the processor completely stalls, before continuing its normal operations. A flowchart of the RTSP procedure is shown in Fig. 5.12. As seen in the flowchart, the PIC first waits for the oscillator to lock before entering this operation. The oscillator should be locked at all times during operation, but clock variation can occur due to power dips and other hardware
failures. The fragile nature of this process requires that the PIC should be completely stable during the operation. Note that the page used to implement this virtual EEPROM must be chosen at an address above that of the actual program code. The page boundaries are 400h apart and the table offset should therefore be a multiple of that.

![Program Memory Organization](image)

**Figure 5.11:** *Program memory organization taken from [Mic07]*

The flash memory update option can be executed from the GUI, or from the base station via RF. The number of flash memory writes is limited and they should only be used when important settings are changed. The complete memory map of the settings stored with RTSP during a memory update procedure is shown in Table 5.1. Note that the upper byte (byte 3) is not used, as explained by Fig. 5.11. These settings include the complete RF configuration register.

![Flowchart of the RTSP Procedure](image)

**Figure 5.12:** *Flowchart of the RTSP procedure*
of both the RF transceivers. All the ADC settings required to initialize the ADC ports needed on start-up are also stored. The other settings include all the network parameters required to ensure stable communication. The most important value set in this table is that of \textit{Progcnt}. This variable is a counter that indicates the number of times a specific page has already been written to. When the PIC starts up it tests this first word of the page to see whether it contains any previously stored data. If the counter is greater than 0, the PIC initializes itself according to these settings, otherwise it is initialized according to the default manufacturer’s settings.

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Progcnt</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>2</td>
<td>BoardID[0]</td>
<td>BoardID[1]</td>
<td>BoardID[2]</td>
<td>0x00</td>
</tr>
<tr>
<td>4</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>8</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>10</td>
<td>nRF1[0]</td>
<td>nRF1[1]</td>
<td>nRF1[2]</td>
<td>0x00</td>
</tr>
<tr>
<td>16</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>18</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>19</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>20</td>
<td>nRF2[0]</td>
<td>nRF2[1]</td>
<td>nRF2[2]</td>
<td>0x00</td>
</tr>
<tr>
<td>26</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>28</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>30</td>
<td>ADC[0].INFO</td>
<td>ADC[1].INFO</td>
<td>ADC[2].INFO</td>
<td>0x00</td>
</tr>
<tr>
<td>32</td>
<td>ADC[3].INFO</td>
<td>ADC[4].INFO</td>
<td>ADC[5].INFO</td>
<td>0x00</td>
</tr>
<tr>
<td>34</td>
<td>ADC[6].INFO</td>
<td>ADC[7].INFO</td>
<td>ADC[8].INFO</td>
<td>0x00</td>
</tr>
<tr>
<td>36</td>
<td>ADC[9].INFO</td>
<td>ADC[10].INFO</td>
<td>ADC[11].INFO</td>
<td>0x00</td>
</tr>
<tr>
<td>38</td>
<td>ADC[12].INFO</td>
<td>ADC[13].INFO</td>
<td>ADC[14].INFO</td>
<td>0x00</td>
</tr>
<tr>
<td>40</td>
<td>ADC[15].INFO</td>
<td>ADC[16].INFO</td>
<td>ADC[17].INFO</td>
<td>0x00</td>
</tr>
<tr>
<td>42</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>44</td>
<td>TTL</td>
<td>BO</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>46</td>
<td>Scan_TTL</td>
<td>RTCC_interval</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>Borate</td>
<td>NumRetallow</td>
<td>NumScanRetallow</td>
<td>0x00</td>
</tr>
<tr>
<td>50</td>
<td>ServiceRoutine</td>
<td>DropPeriod</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

Table 5.1: Memory map of the RTSP settings

5.6 TX RF data

RF transmission can be triggered by numerous sources. Once the completed data packet is sent to the TX procedure the processing is exactly the same, independent of source. The basic flowchart for transmitting RF data is shown in Fig. 5.13. This procedure is the same for both transceivers. The RFx therefore, denotes either RF1 or the RF2 module. Whenever data arrives at the TX procedure, it first tests to see if the transmitter is available or not. If the transmitter is not available, the data is stored in the software TX buffer. A dedicated buffer does not exist
and data is added to the ACK stack with the retransmission count set to 0. This indicates that it has not been transmitted yet and is assumed to be in the buffer. As explained in Section 5.4.1, the transmitter’s availability is indicated by the TX flag. This flag is cleared after the data has been transmitted or when the data in the nRF905 TX buffer has not been transmitted for longer than some pre-defined time (TTL). If the transmitter is available, the type of message to be transmitted is tested. If the data is either a command (CMD) or a message (MSG) it is added to the ACK stack before sending it via SPI to the RF transceiver. Acknowledge packets (ACK) are sent directly to the RF transceiver. Before data can be transmitted, the channel must be sniffed to determine whether it is clear or not. The channel is busy whenever the carrier of the same RSB is on (used for antenna testing purposes) or if another RSB is busy transmitting (the CD pin is set). If the channel is busy, the packet to be transmitted will be set to go into back-off mode.

5.7 RX RF data

The RX procedure for RF data is very complex. The RX process is different from the TX process in the sense that it only receives data from the RF interrupt. The basic flowchart for this process is shown in Fig. 5.14. Note that the RX flag test is only a precautionary method used to prevent data from being overwritten, which should not happen, however, because all RF interrupts are disabled on data reception and are only enabled after the data has been processed (RX=0). On data reception, all packets are split into their respective message types. If the received MSG packet is destined for this RSB, it is logged and a ACK2 message is sent back to the source RSB. If the data has to be routed through this RSB to its destination, it is forwarded and an ACK1 message sent back to the previous station, while all other MSG packets are dropped. CMD packets destined for this station are executed and an ACK2 message sent back. CMD packets that have to be routed through this station are forwarded and an ACK1 message sent back to the previous station that sent this packet, while all other packets are dropped. The choices surrounding ACK messages are shown in Fig. 5.14. Note that this flowchart is only the basic chart and does not include all options. The scan packets in particular are only briefly explained.

5.8 Scan procedure

The basic network scanning procedure will be discussed in this section. For a full description refer to Chapter 3. As explained in Chapter 3, every scan sequence consists of a scan for both children and grandchildren. The children scans can be split into scanning for existing children (children already registered in the children list) and scanning for new children, also referred to as orphans. The term grandchild is used to refer to any sibling of a parent station, independent of the number of hops between the parent and the grandchild station. The scan for children
Figure 5.13: Flowchart of the TX process

The TX flag shows that the nRFx TX buffer is currently in use and not available. This is done to prevent data from being lost if the transmitter goes into BACKOFF.

The TX flag is cleared when a DR interrupt is triggered by data transmission, showing that the data has been transmitted and the buffer is available.

The last SPIx write time is used to release the transmitter (nRFxflags.TX = 0) after a period of time, if the transmitter is not used for a long time (TTL). This is done to prevent the TX buffer from overflowing if the transmitter is constantly unavailable due to some carrier being present.
Figure 5.14: Flowchart of the RX data process
procedures will be discussed first.

The flowchart in Fig. 5.15 shows the basic flow of data within the Scan_Network procedure. This procedure is called whenever a scan for children must be executed. Before scanning for children, it must be ensured that the station performing the scan has a valid level. An un-routed station cannot perform a scan. Another important test that must be performed is to determine whether a new scan must be started, or if a current scan should be continued. The Set_RTCC_alarm procedure is used to set when the next scan should occur. This time of next scan value is sent to all the stations during scanning to ensure that scan synchronization is kept.

The Scan_REQ procedure is used to process SCANNET packets received at a station. The flowchart for this procedure is shown in Fig. 5.15. If the scan packet is only used to detect whether a child is still available, only an ACK is sent back to the parent as confirmation. If the child receiving this message is not currently a child of the station sending this message, it sends its own ID and level, embedded into an ACK2 message, back to the parent station. This only happens if the new link would be superior to an existing link.

![Flowchart of the Scan_Network and Scan_REQ procedures](image)

Figure 5.15: Flowchart of the Scan_Network and Scan_REQ procedures

The Scan_Received procedure is used whenever an ACK message is received in reply to a SCAN- NET or SetLVL command. The flowchart of this procedure is shown in Fig. 5.16. The basic task of this procedure is to determine the next scanning step, depending on the received data.
Figure 5.16: Flowchart of the Scan_Received procedure

If data is received from an existing child, the next child should be searched for. If an ACK2 is received in reply to a SCANNET packet from an orphan station, a SetLVL command is sent to this station, while this new station is added if the ACK2 received is in reply to a SetLVL command. If a new station is added, the station would scan for more orphan stations.

The GP scan is used to allow the children to scan for new children. The GP scan process is managed by the Scan_GrandChildren procedure. The basic flowchart of this procedure is shown in Fig. 5.17. The first step before executing a GP scan is to test whether a new GP scan should be initiated or if a current scan should be completed. The GP scan procedure does not look for new children like the scan procedure, but is used to allow children to look for children. As a result, only stations with children will execute this procedure. This procedure follows exactly the same procedure explained in Chapter 3 and will, therefore, not be repeated.

The last important block, when it comes to scanning, is that of the service routine. Only a small excerpt from this procedure will be discussed. A flowchart of this excerpt is shown in Fig. 5.18. The scan for new children is finished when a scan packet (SCANNET) is dropped. When this happens, the station would assume that no new stations are available, and would, therefore, start the GP scan. The GP scan is stopped when the last FINSCAN packet is received. Errors could occur when an ACK message is not received for a GPSCAN packet. When this happens the packet would be dropped, assuming that the child requested to do the scan is no longer functional. If this happens, the next child would be asked to perform a GP scan.

The complete scanning algorithm is explained in Chapter 3. The flowcharts and explanations given in this section are used only to indicate briefly how the scanning and algorithm was
Chapter 5 — Embedded PIC software

5.9 Summary

All the processes of the embedded PIC software are interrupt driven. This makes the use of good interrupt handling procedures essential. All the network parameters and network maintenance settings can be set in either the GUI, or by the base station via the RF link. All these settings can be stored in the virtual EEPROM, created with the use of RTSP. The program is prevented from stalling by the use of a WDT. The ADC settings are also set either in the GUI, or by the base station via the RF link. These ADC settings allow the PIC to completely initialize its ADC ports, even after a power failure. The ADC info field is used to uniquely specify the type of each transducer used. This field is used by the server to identify the type of measurement acquired. The scan procedures utilized to keep the routing paths up to date have been briefly discussed in this chapter, but a more comprehensive review is available in Chapter 3.

Figure 5.17: Flowchart of the Scan_GrandChildren procedure

implemented in the software.
Figure 5.18: Flowchart of an extract from the service interval procedure, which influences the scan process
Chapter 6

Server Software (GUI)

This thesis revolves around the design of a data acquisition (DAQ) network. DAQ networks require the use of a server to log the measured data. To enable a user to set up the settings of the stations and to control them, the user must be able to connect to it somehow. This interface and the server are added together in one program. To simplify the complexity of the interface for the user, the program is designed to have a graphical user interface (GUI). With this GUI the user can easily manage the network by simply clicking buttons, without knowledge of how the actual interface works. The design of this GUI is described in this chapter.

6.1 Basic design

The GUI was designed with the Borland Delphi software. Delphi is a very powerful program most suitable for rapid application development (RAD), using a visual interface. RAD was initially developed by James Martin in 1991. With RAD, prototype procedures are created as objects which can easily be incorporated into new projects. These objects speed up the design process enormously. These objects are programmed using the Delphi language, an object type Pascal. New components can also be created or downloaded from the internet. The Delphi user base is very large, offering extensive support and countless available objects. Another advantage of Delphi is that it puts a strong emphasis on database connectivity. This allows for easy integration with databases, which can be used for logging purposes.

The GUI was designed to perform numerous tasks. These tasks can be subdivided into four main categories: general tasks, network maintenance, data logging and performance analysis. These categories will now be discussed individually.
6.2 General tasks

The general tasks include the serial interface and the ADC settings. The serial interface is the essence of the GUI and all settings and logging are performed via this interface. The serial connection can be established in the Main tab of the GUI. Different COM ports are supported. The ADC settings control which ports should be monitored and also define what type of transducers are connected to these ports. These settings can be set either directly by a PC connected to a RSB or by the server via the RF link. These settings are available under the ADC settings tab.

6.2.1 Serial connection

Before the GUI can perform any major tasks, it must be able to connect to a RSB. This connection is established via a serial interface. An open source component, ciacomport (downloaded from [cia06]), was used to implement this serial interface. The same serial settings were used as explained in Section 5.4.2. Note that the end sequence is repeated twice when data is sent to the PC (#10#13#10#13). All data is sent to the server via this interface. A command character set is used to distinguish between the different types of messages. The first byte of a received packet always contains this command character. In some cases, the second byte is used as a subclass descriptor. The complete implemented character set is shown in Table 6.1. The PC and RSB columns are used to indicate whether this message is sent (S) from or received (R) by the PC and RSB respectively. Note that the packet layout for the sent and received packets is not necessarily the same.

The layout of some of the important commands will now be discussed. The fact that the end sequence has to be added to the packet will not be stated every time, but has to be remembered. The connect signal sent from the PC contains only the command, while the packet sent from the RSB is much more complex. The connect packet sent from the RSB contains all the RSB settings to allow the GUI to display them. This allows the user to immediately know all the network settings and all the other required information regarding the connected RSB. These settings include the RSB ID, level, parent, AP, unique ID code, TTL, BO, service routine period, scan period, drop period and the number of retransmissions allowed.

The update flash option (U) informs the connected RSB to write its current settings to the flash memory using RTSP. The carrier settings (t) are used for antenna testing purposes. When a carrier is enabled, it transmits a constant carrier at the selected frequency. Note that the subclass descriptors with this command are 1, 2, 3 and 4 respectively. The last scan command (h) sends the time of the last scan sequence performed to the PC. This last scan time can be used together with the the scan settings to determine when the next scan should occur.

The measured ADC settings (m) sent to the PC, contain the complete standard RF packet
Table 6.1: Serial command set

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
<th>PC</th>
<th>RSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>Connect serial</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>d</td>
<td>Disconnect serial</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>a</td>
<td>Set as base station</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>l</td>
<td>Change Board ID</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>U</td>
<td>Update flash</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>x</td>
<td>Read all the nRF1 registers</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>X</td>
<td>Read all the nRF2 registers</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>k</td>
<td>Read 1 nRF1 register</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>K</td>
<td>Read 1 nRF2 register</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>p</td>
<td>Change 1 nRF1 register</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>P</td>
<td>Change 1 nRF2 register</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>v</td>
<td>Send data via RF</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>t</td>
<td>1 - Enable RF1 carrier</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>2</td>
<td>Disable RF1 carrier</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>3</td>
<td>Enable RF2 carrier</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>4</td>
<td>Disable RF2 carrier</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>s</td>
<td>Set RTC</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>S</td>
<td>Read RTC</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>h</td>
<td>Last scan hh:mm:ss</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>n</td>
<td>Children’s names</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>g</td>
<td>Grandchildren’s names</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>A</td>
<td>R - Read ADC of RSB connected to PC</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>r</td>
<td>r - Read ADC of a remote RSB</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>S</td>
<td>S - Set ADC settings of RSB connected to PC</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>s</td>
<td>s - Set ADC settings of a remote RSB</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>m</td>
<td>Measured ADC values sent to PC</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>F</td>
<td>Send emulated ADC packet to AP (PERF)</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>e</td>
<td>Emulated packet sent</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>E</td>
<td>ACK received for emulated packet</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>b</td>
<td>Emulated ADC packet received at BS (if logging enabled)</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>C</td>
<td>A - Configure RTC_interval, DropPeriod and ScanPeriod</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>r</td>
<td>r - Software RESET</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>T</td>
<td>T - Configure TTL</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>s</td>
<td>s - Configure Scan_TTL</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>B</td>
<td>B - Configure Borate (BO TTL)</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>b</td>
<td>b - Configure BO</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>R</td>
<td>R - Set ServiceRoutine period</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>N</td>
<td>N - Configure NumRetAllow</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>S</td>
<td>S - Configure NumScanRetAllow</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
<tr>
<td>y, z, Z</td>
<td>Debugging must be enabled on RSB</td>
<td>$+$R</td>
<td>$+$R</td>
</tr>
</tbody>
</table>

described in Chapter 3. From this packet, the GUI can extract all the information needed to uniquely identify the measurement. The information used to identify this are the source ID, source level, source ID code, destination ID and the packet number. A time stamp is also added by the GUI. These measurements are written to a database as explained in Section 6.4.

The serial connection and all the settings gained from the connection procedure is displayed in the Main GUI tab, shown in Fig. 6.1.

### 6.2.2 ADC settings

The ADC settings tab, shown in Fig. 6.2, controls which ADC ports must be enabled for monitoring. The drop down box can be used to select the type of transducer connected to that port.
Chapter 6 — Server Software (GUI)

Figure 6.1: A screen capture of the Main tab of the server GUI

An extra column is also available for an additional description. New transducer types can be added under the PROBES tab of the Logs tab. The different types of probes used are stored in the PROBES record of the DATA.mdb database. The probes can also be edited manually in this file. Note that the conversion column is used to scale values if needed. The ADC settings can be read from or written to a RSB connected directly to this PC. If the REMOTE option is used, the settings of a RSB connected via a RF link through the AP board can also be read or changed. The logging of the data is described in more detail in Section 6.4.

6.3 Network maintenance

The network maintenance tasks consist of setting up the configure registers of the RF controllers, the network parameters and managing all scanning procedures. These parameters can be set via the ADVANCED SETTINGS and NETWORK sections of the Main tab shown in Fig. 6.1. The ADVANCED SETTINGS section contains all the configure settings (C) given in Table 6.1. The NETWORK section can be used to start a new scan sequence if the connected RSB is the BS. The connected RSB can also be set as BS from within this NETWORK section.

Another task of the network maintenance block is to keep the list of stations in the network up to date. After every completed scan sequence (scan for children and grandchildren), the
BS sends the names of all its children to the server, before requesting a list of children from every other station. Whenever the server receives a list of children from the BS, it clears the complete children list and adds the new children. When a list of children is received from any other station, the server simply adds the names to the list. Before a station is added to the list the server will first check its current list to see if that station is not already in the current list. The flowcharts of this list population procedure are shown in Fig. 6.3. This child list contains the child’s name, level and unique ID.

After receiving a child list from a station, the server updates the graphical representation of the network layout. A screen capture of the Network layout tab is shown in Fig. 6.4. In this figure the station called APB1 is the BS, JAK2 is a level 2 station and EIK2, LHK1 and RSB7 are all level 3 stations routing through JAK2. This schematic representation greatly facilitates the visualization of the network topology.

### 6.4 Data logging

The primary task of the server is to log the measured ADC data. The easiest and most structured way of logging data is to use a database. Microsoft Office Access was used to create the database. Delphi supports the use of ActiveX Data Objects (ADO). ADO was created by Microsoft and
can be used to access and manipulate data from various sources using an OLE DB provider. The OLE DB platform was also created by Microsoft. The ADO connection provides a layer between the object Pascal code and the OLE DB, which is used to access data stores. With this ADO connection no knowledge of SQL is needed to manipulate the database, but SQL queries can be executed if required. Data filters are implemented using standard SQL queries. The flowchart of the GUI startup procedure is shown in Fig. 6.5. Note how the initialization procedure is only used to link all the different ADO connections to the respective tables. The database used for logging consists of five basic table formats. These tables are named: TEMPLATE, SETTINGS, PROBES, Emulate and BaseEmulate. The function of each of these tables will now be briefly discussed.

The PROBES table contains a list of all the different ADC probes used and their respective conversion factors. These conversion factors are used for scaling of measured values, if needed. New probes can be added in the PROBES tab of the Logs tab, or it can be changed manually in the DATA.mdb database file. The Emulate and BaseEmulate tables are used during performance analysis of the network, and are explained in more detail in Section 6.5.

The SETTINGS table contains a list of all the stations that are monitored, together with their respective ADC settings. These settings include the ADC_USE variable which indicates which ADC ports are monitored. The ADC_USE variable is a 16 bit variable, with each bit representing the corresponding ADC port. A set bit represents a used port. The ANx field specifies the probe type connected to the ADCx port. The SETTINGS table is automatically updated every time an ADC settings packet (AR or Ar) arrives at the server. A flowchart of the ADC adjust procedure is shown in Fig. 6.6.

The data of each monitored station is kept in a separate table, which has the same name as that station. These tables are initially created by copying the TEMPLATE table and then just renaming the new copied version to the name of the RSB. These tables are created whenever the

**Figure 6.3: Flowchart of the children logging procedure**
ADC SETTINGS table are updated, or if the table does not yet exist, but the station’s name is in the SETTINGS table. The logged data consists of 7 descriptive columns, which can be used to uniquely identify each log, and 3 columns for each of the 16 ADC ports. Even though only 9 of the 16 ports are available, the other ports are also added to allow for future changes. The 10 different column types are:

- **NO** - A unique number used by the database as a key field. Each record in the database has a completely unique number.

- **Time** - A timestamp added by the server to indicate the exact time of reception of the data. Note that the time field indicates when the server received the message and not when the RSB sent the message. The difference between these two times can be neglected because the typical transmission time will be less than a second. A timestamp is not added to an ADC log by the station, which reduces the packet size. The timestamp added includes both the date and time, and is stored in the long data format (yyyy/mm/dd hh:mm:ss).

- **SID** - The name of the source RSB.

- **SLVL** - The level of the source RSB.

- **DID** - The name of the destination BS. This is added because multiple base stations can exist in one network.
Figure 6.5: Flowchart of the GUI startup (initialization) procedure

- CODE - The unique routing ID of the source RSB. See Chapter 3 for more detail on how this code is assigned.

- PN - The packet number. This column can be used to prevent multiples of the same packet to be added to the log file.

- ANx - The x is used to refer to any one of the 16 ADC ports. The ANx column are used to store the value of the received data of the corresponding ADC port.

- ANx,Type - This column contains the probe type used. The value stored in this field is extracted from the SETTINGS table.

- ANx,Description - This column contains an additional description of the specific probe type. The value stored in this field is extracted from the SETTINGS table.

The flowchart of the data logging procedure is shown in Fig. 6.7. The flowchart clearly shows that the SETTINGS table is tested before any data is added to determine whether the source station is in the monitored list. If the station must be monitored, the server also tests whether a table exists with this RSB’s name. If a table does not already exist, the server will create a new table. This test might seem redundant, seeing that the table was already created during the ADC_adjust
procedure explained above. This feature is added to allow the server to automatically fix itself if a table should become corrupted or deleted. It must be noted that the previous value of a port must always be remembered. An ADC packet only contains the values of ADC ports that have changed. When data is added to an existing table, the server must always use the previously logged data to acquire the values of those ports that did not change.

The Logs tab of the server allows the user to extract data from the database. The user can either use the built in query options to extract data, or can set up his own queries. The built in functions supply the names of all the stations, the probes and the port numbers. Combinations of these three criteria can be used to set up a useful query. More advanced users can use the manual query option to extract data. The extracted values are displayed in the LOGS tab of the Logs tab. It is much easier to analyze data if it is displayed graphically. A graph utility component, called Teechart, was added to the Delphi code to create a graphical environment. This new environment allows the user to display the extracted data graphically in the GRAPHS tab of the Logs tab. This graphical environment supports all the basic features such as zooming, panning and refreshing of data. A screen capture of a typical graph is shown in Fig. 6.8. As

Figure 6.6: Flowchart of the Adjust_ADC procedure
explained earlier, the timestamp includes both the date and time. If the displayed graph contains data from more than one day, the GUI automatically shows the date and not the time. When zooming in, the date would change back to a time. Note that the data to be displayed can be selected in the lower left corner.

### 6.5 Performance analysis

The performance of the network can also be analyzed by the GUI. The performance tests are described in detail in Section 8.2. Each station to be analyzed has to be connected to a PC. The Emulate table is used to store performance packets (PERF). The flowchart of the performance logging procedures is shown in Fig. 6.9. The Difference column represents the latency of each packet, and is determined by the difference between the time when the performance packet was sent and when the ACK for the corresponding message was received. The effective data rate
can be determined from this value. The DATA_EC variable is stored in the payload section of
the RF packet when data transmission occurs. This variable is used to count the number of
packets sent from this station. The packet number (PN) embedded in the RF packet header
is limited to a maximum count of 255 packets, and is only used to test for duplicate packets
received within a very short time of each other. The data received by the BS can also be logged
if necessary. This data is stored in the BaseEmulate table. The same graphing utility used to
display the measured results of the ADC logs is used to display the latency and data rate for a
station.

Figure 6.8: Screen shot of the GRAPH tab of the Logs tab

The server has a save option, which can be used to save the data from the Emulate table to
another table. This allows the user to save the results from different network tests. The Emulate
table can also be cleared whenever desired. SQL queries can also be used to extract data from
previously stored tables. The graphs obtained can be printed or exported. The graphs can be
exported as bitmaps (BMP), metafiles (WMF), enhanced metafiles (EMF) or as teechart files
(TEE). The teechart files can be opened with a freeware program TeeChartOffice, for further
editing if needed. A screen capture of the GRAPHS tab is shown in Fig. 6.10.
6.6 Error handling

The GUI was created to automatically handle all types of error. The try...except statement are used throughout the program. This procedure prevents the station from crashing and merely reports all errors. Minor infractions are reported in the status bar, at the bottom of the GUI, while major errors are reported via an error dialog. This includes run-time errors and data validation. Whenever an error occurs, the GUI would extract the error message directly from the error handler and would, therefore, have complete knowledge of the actual problem.

The GUI was designed to run on the Microsoft Windows operating system. The GUI also requires the qtintf70.dll run-time library. This library must be installed on all PCs without Delphi installations. The file can be downloaded from the internet, but is also supplied on the accompanying multimedia disk.
6.7 Summary

The basic control and logging features of the GUI has been described in this chapter. The control features include settings such as inter-scan times, RF configuration, ADC settings and general network settings. The GUI can be used to manipulate settings of stations either connected directly to the PC via a serial cable, or connected via an RF link.

All ADC measurements sent to the BS are logged in a database, which is managed by the server GUI. The logged data includes both ADC data and performance analysis data. A complete graphical interface was added to display the measured results. Graphs enhance the visual interface and enable quick analysis of data. The basic filters supplied by the GUI allow the user to quickly filter for desired data, while the advanced manual SQL capabilities allow advanced database users to manipulate and access data with ease. The GUI is also completely capable of saving and restoring data, which makes it even more versatile. The database can easily be copied and data transferred to other PCs. The GUI is available on the accompanying multimedia disk.
Chapter 7

Performance Predictions

7.1 Introduction

When designing a new network, there are many things that have to be taken into consideration. One of the most important decisions to be made is that of which communication strategy and protocol to use. There are many different kinds of protocol, each with its own advantages and disadvantages. Some of the main factors in choosing the optimal protocol are the physical layout of the network, required data rate, transmission range, reliability and cost. In this chapter the protocols considered will be discussed and analyzed. This is done using an analytical approach as well as a direct approach in the form of a Simulink model.

The mathematical analysis of two types of protocol is discussed, that of collision free protocols and contention protocols. The collision free protocol, Round-Robin Polling, will be analyzed first using two different approaches. The first approach uses standard analytical mathematics and is referred to as the conventional method, while the second method is based on queueing principles. Lastly the CSMA contention protocol will be discussed and analyzed, using queueing theory.

The CSMA contention protocol is also modeled using Simulink. The design of this model will be discussed and the results compared with those of the mathematical model. A full comparison of the results of the Simulink model, the mathematical model and the actual measured results will be discussed more thoroughly in Chapter 8. Before starting with the mathematical analysis, a brief overview of queueing theory will be presented, as it plays an integral part in both of the analyses.
7.2 Queueing Theory

7.2.1 Background

Queueing theory is the mathematical study of waiting lines, better known as queues. Agner Krarup Erlang published the first paper on queueing theory in 1909. Since then many people have contributed to this topic. David G. Kendall introduced the $A/B/m/k$ queueing notation in 1953, where $A$ describes the arrival process, $B$ describes the service process, $m$ is the number of servers while $k$ is used as a sub-class descriptor. This notation is still the preferred way to describe a queueing system. The most common case is the $M/M/1$ queue, where $M$ denotes a Markov process with exponential interarrival times. This is one of the easiest queueing systems to analyze. Beside Markovian (M) other common distributions also exist, such as, General (G), General Independent (GI), Erlang (E), Hyper-exponential (H) and Deterministic (D). The general case is very difficult to analyze. The deterministic case implies that all intervals are the same (periodic).

7.2.2 Basic Queueing Principles

The basic model used to analyze queues is shown in Fig. 7.1. Every queue consists of an input from some source, the queue itself which has some characteristics and an output to some server or other process. Every queue can be characterized by 6 basic parameters (See [RW06]). These parameters are:

- **Source population**
  The source population describes the physical source to the queue. The number of stations in this network is limited, while the number of events created at each station is unlimited. If all messages were to be sent individually, the source population would therefore be infinite. If an assumption is made that all data in a station’s transmit buffer is transmitted as one packet, the source population is finite. For the finite case it should be held in mind that if a station has data to transmit, and is therefore waiting in the queue, one less station can generate new data, whilst if a station’s data is serviced by the server, and therefore removed from the queue, that station is now added to the other stations that can generate new data.

- **Arrival rate and distribution**
The arrival rate describes the rate at which new entities arrive at the queue. For the infinite source case the arrival rate is constant while for the finite source the arrival rate is a function of the number of idle stations. If a station is already in the queue, it cannot produce new entities and the arrival rate for that station is therefore 0. There are various PDFs describing the arrival process, and therefore the analysis for each queue is scenario specific.

- **Service rate and distribution**
  The service rate describes the rate at which entries are taken from the queue and serviced. The service rate must be greater than the arrival rate, otherwise the queue will grow boundlessly. The PDF of the service rate is also network specific. With the protocol in use, the packet size is constant. The service rate can therefore be seen as a deterministic process.

- **Number of servers**
  Even though each station can be seen as a server to all other stations in the routing tree below it, there is only one true server, and that is the base station. All data has to be processed by this station, and therefore all data will have to be queued up before this station. If multiple servers are used, the system latency would decrease enormously.

- **Queueing discipline**
  The queuing discipline describes the process governing the order in which items in the queue are serviced. Different schemes include first come first served (FIFO), last in first out (LIFO) and prioritized queues. The discipline used in the network studied is FIFO. This is based on the fact that the oldest messages should be serviced first to try and reduce the amount of retransmissions generated due to expired TTL counters.

- **Queue buffer**
  The queue size has a big influence on network stability. If the queue is full and new entries arrive, the new entries would be dropped. This would generate new entries due to retransmission, which would therefore increase the traffic density and packet latency. For a finite source population a queue with length equal to or greater than the population size will always be stable, regardless of the inter arrival or service rates. This, however, is not the case for an infinite source population, which would require an infinite queue size to ensure stability. If the service rate is lower than the combined inter arrival rate, the queue would grow infinitely and thus be unstable.

An important queueing principle that must be mentioned is Little’s theorem. This theorem was derived by J.D.C. Little in 1961 and states that the average number of entities ($N$) in a steady state queueing system is always equal to the product of the average arrival rate ($\lambda$) and the average time ($T$) a customer spends in the system. This theorem is applicable regardless of the arrival distribution, the service distribution, number of servers or the queueing discipline. The only prerequisites are that the system must be stable and that the queue buffer space must be
greater than the average queue length. In symbols this equation can be expressed as in eq. 7.1. If a system is of the form $M/M/1$, the total number of entities in the system can be determined using eq. 7.2, where $\rho$ is the utilization of the network. This derivation is based on equilibrium states and is derived in [RW06]. Utilization is described in Section 7.2.3. If eq. 7.1 and 7.2 are combined, and it is remembered that $\rho = \frac{\lambda}{\mu}$, the total time spent in a memoryless queue can be derived to be that stated in eq. 7.3.

$$N = \lambda T \quad (7.1)$$

$$N_q = \frac{\rho}{1 - \rho} \quad (7.2)$$

$$T = \frac{1}{\mu - \lambda} \quad (7.3)$$

### 7.2.3 Modeling the Queueing System

Modeling of queues is usually based on Markov chain theory (See: [Nic06], [Hoc97] and [RW06]). A Markov process is memoryless, which means that future arrivals are independent of previous arrivals. This memoryless property states that if an arrival process has interarrival times of 10 seconds, and an arrival occurs after 5 seconds, this does not mean that the next arrival will occur after 15 seconds, seeing that the process is memoryless and cannot remember when the previous arrival occurred. This memoryless property enables us to analyze a network according to the statistical properties of the different processes, ignoring prior activities.

Another important measure in queueing networks is that of the utilization. Utilization ($\rho$) is a measure of how busy the server is. It can be described as the time the server is occupied, in comparison to the total time available. In [Hoc97] the relationship for utilization is derived and the final result shown in eq. 7.4 is obtained. It must be remembered that the server cannot process more data than its maximum capacity and, therefore, the utilization can never be equal to or greater than 1. If the arrival rate is greater than or equal to the service rate, the queue would grow boundlessly and hence the network would be unstable.

$$\rho = \frac{\lambda}{\mu}, \rho < 1 \text{ to ensure stability} \quad (7.4)$$

The arrival process is often modeled using a Poisson process. A Poisson process is described by [Hoc97] as a counting process for the number of randomly occurring events taking place in a certain time interval. The distribution of a Poisson process is given in eq. 7.5. This equation gives the probability that $n$ entities will arrive during time $t$ for an arrival rate of $\lambda$ entities per unit time.

$$P_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t} \quad (7.5)$$

This short introduction to queues should be sufficient to complete the mathematical model. If further insight is needed during the derivation, it will be explained in that section.
7.3 Round-Robin Polling (RRP)

7.3.1 Background

With RRP the base station is completely in control. Only the base station can provoke data transmission and therefore the possibility of collisions is removed, hence the class type collision free protocol. With RRP the base station polls each station separately to see whether it has data or not. The base station then waits for a reply from that station. If the station has data, it sends this data back, otherwise it sends a packet indicating that it does not have data. If neither of these packets is received by the base station after a certain time ($t_{TTL}$), the base station would retransmit the poll-request signal. No ACK signal is sent by the base station when it receives the data, seeing that this ACK signal would be subject to the same noise level as the initial messages (See also: The Two army scenario [Wol02] and [Tan96]). Once the base station receives the poll-reply message from a station, it will poll the next station. When finished with polling all stations, the base station would start a new poll cycle.

RRP can be analyzed using two different approaches. The first approach uses normal analytical mathematics and is referred to as the conventional approach. The second analysis is based on queueing theory, as explained in Section 7.2. Before the analysis of these two models can be derived, certain assumptions have to be made. Some of these assumptions are valid for all cases, while others are needed to simplify the model. They are:

- Network is stable and all routing tables already exist.
- Stations in the network are stationary (no mobility). This is a valid assumption, as all monitored stations are stationary.
- No lost links due to bad links or power failures.
- Processing time at the stations and server is constant.
- Packet lengths are fixed, as a result of the transceiver architecture.

7.3.2 Conventional Method

The time needed to poll one level 2 station (no multi-hop) is given by eq. 7.6 where $t_{preq}$ and $t_{prep}$ is given by eq. 7.7 and 7.8 respectively.

$$t_{min} = t_{preq} + t_{prep} \quad (7.6)$$

$$t_{preq} = t_{preamble} + t_{postamble} + t_{rise} + (t_b)(b_{preq}) + t_{prop} + t_{gen} \quad (7.7)$$

$$t_{prep} = t_{preamble} + t_{postamble} + t_{rise} + (t_b)(b_{prep}) + t_{prop} + t_{gen} \quad (7.8)$$

With:
\( t_{\text{min}} = \) Time needed for a complete single poll cycle

\( t_{\text{preq}} = \) The time to send a complete poll-request packet

\( t_{\text{prep}} = \) The time to send a complete poll-reply packet

and:

\( t_{\text{preamble}} = \) Time to transmit the pre-amble

\( t_{\text{postamble}} = \) Time to transmit the post-amble

\( t_{\text{rise}} = \) The time needed by the transceiver to charge up before transmission can begin

\( t_{\text{prop}} = \) Propagation delay

\( t_{\text{gen}} = \) General processing time

\( t_b = \) Time to transmit one bit

\( b_{\text{preq}} = \) Amount of bits in a poll-request packet

\( b_{\text{prep}} = \) Amount of bits in a poll-reply packet

Eq. 7.6 gives the absolute minimum time required to complete one poll cycle and does not take retransmissions or multi-hop into account. This, however, is a definite problem which has to be incorporated into the prediction model. Retransmission is needed when packets are lost due to some form of system noise or interference. If requested data is not received after \( t_{\text{TTL}} \), the base station assumes that the packet got lost and then retransmits the request signal. It should be remembered that \( t_{\text{TTL}} \) must be longer than \( t_{\text{min}} \) to prevent data from being retransmitted if the initial packet is still in the system and on its way back to the base station. The number of retransmissions needed to transmit a packet cannot be determined physically due to the unpredictable nature of system noise. As explained by [Wol02] the number of retransmissions due to noise can be modeled by assuming that a certain percentage of the packets will have to be retransmitted. Eq. 7.6 can now be expanded to eq. 7.9, where \( \delta \) is the fraction of the packets which are to be retransmitted. This equation however does not take into account that some packets will have to be retransmitted more than once. With the previous equation, if 100\% of the packets were to be retransmitted, the latency will only be \( t_{\text{TTL}} + t_{\text{min}} \), which is clearly not the case. Eq. 7.9 is now expanded to eq. 7.10, which also includes the capability to do multiple retransmissions. To derive this equation, \( \delta \) can be thought of as the probability that a packet will be retransmitted. The probability for one retransmission is therefore \( \delta \), two retransmissions would be \( \delta^2 \), and for \( k \) retransmissions it would be \( \delta^k \). The multiplication factor \( k \) indicates the maximum number of times each packet is allowed to be retransmitted. It should be remembered that \( \delta \) must be smaller than 1 to ensure that eq. 7.10 will converge. The result obtained is meaningful if it is taken into account that for 100\% retransmission (\( \delta = 1 \)), the latency would
be infinite, while according to the equation derived in [Nic06] it would only be $t_{\text{min}} + t_{\text{TTL}}$.

$$t = t_{\text{min}} + (\delta)(t_{\text{TTL}})$$  \hspace{1cm} (7.9)

$$t = t_{\text{min}} + \sum_{i=1}^{k} \delta^{i} (it_{\text{TTL}})$$  \hspace{1cm} (7.10)

The case studied till now is for a single-hop system. The architecture under investigation is clearly a multi-hop system. By looking at Fig. 7.2, it can be seen that if the base station were to request data from a level 5 station, it would require 8 hops in total. This, however, is only 4 poll hops (4 poll-request signals and 4 poll-reply packets), which is exactly 4 times the number of a normal level 2 request cycle. One $h$ hop station (level $h + 1$) can therefore be replaced by $h$ level 2 stations as seen in eq. 7.11. This equation gives the average time it would take the base station to read data from a $h$ hop station.

Figure 7.2: Multi-hop system

$$t_{h} = (h)(t)$$  \hspace{1cm} (7.11)

If there are $N$ level $h + 1$ stations (and only these stations) the time it would take the base station to service all stations would be given by eq. 7.12.

$$t_{N,h} = (N_{h})(t_{h}) = N_{h}h \left( t_{\text{min}} + \sum_{i=1}^{k} \delta^{i} (it_{\text{TTL}}) \right)$$  \hspace{1cm} (7.12)

For a complete network, the above model has to be expanded to a multi-level combined system model. To obtain the total cycle time, the individual cycle times of all levels have to be added. The number of stations in each level plays an important role in the final result. The average cycle time for the base station servicing the whole multi-level network, with consideration of additive burst and white noise, is therefore given by eq. 7.13, where $K$ is the highest level station present in the network and $N_{h}$ is a vector containing the number of stations in each level.

$$t_{\text{cyc BS}} = \sum_{h=1}^{K} N_{h}t_{h}$$  \hspace{1cm} (7.13)

The model can now be simplified by introducing a concept where the $N_{h}$ vector is transformed into a virtual number of stations, $N_{\text{virtual}}$. This is done by multiplying the number of stations
in each level by the number of poll hops required to service one station on that level. This assumption is based on the fact that it would take the same time to service one level 3 station as to service two level 2 stations. Given the fact that a packet sent from a level 3 station has to hop twice to get to the base station, the probability of that packet being lost is 2δ. This is the same as the probability that one packet will get lost if two level 2 stations transmit data.

When a level 2 station has to retransmit a packet, it will only take an extra $t_{\text{min}}$ seconds, while a level 3 station retransmit would take an extra $2t_{\text{min}}$ seconds. Seeing that there are two stations replacing one station, the number of arrivals will also double, because the arrival rate per station stays unchanged. This will cause $2 \times 2\delta$ level 2 retransmissions, which is the same as $2t_{\text{min}}$. This expected latency effect of both these scenarios is the same and the assumption is therefore valid, as stated in eq. 7.14. The final result is given by eq. 7.15. The average time a station would have to wait to be serviced if new data arrives at that station, is given by eq. 7.16. The reason for dividing by two, is that if new data arrives just after a station has been serviced, the station would have to wait $t_{\text{cyc}}$ seconds before it would be serviced again, while if data arrive just before a station is serviced, it would not wait at all. If the average of these two scenarios is taken, the result would be that of eq. 7.16.

$$N_{\text{virtual}} = \sum_{h=1}^{K} ih_i$$  \hspace{1cm} (7.14)

$$t_{\text{cyc}} = (N_{\text{virtual}})(t) = N_{\text{virtual}} \left( t_{\text{min}} + \sum_{i=1}^{k} \delta^i (it_{\text{TTL}}) \right)$$  \hspace{1cm} (7.15)

$$t_{\text{wait}} = \frac{t_{\text{cyc}}}{2}$$  \hspace{1cm} (7.16)

The equations derived were solved using MATLAB. Different noise levels were used and the results compared. The results obtained are shown in fig. 7.3, 7.4 and 7.5. The first figure clearly shows the effect noise has on the system, especially in a network with a higher station count. The second figure accentuates the effect that multi-hop has on the system latency. In this figure, the number of stations indicated on the x-axis is only on the level indicated in the legend, while only one station is present on the lower levels. The last figure clearly shows the snowball effect of noise. As seen, the latency will grow boundlessly for high error rates. It is also clearly visible that noise will affect multi-hop stations much more severely than lower hop stations.

The settings used in this simulation are given below. To determine $t_{\text{prop}}$ it is assumed that the distance between all stations is 1 km. The number of preamble bits and the data rate ($BR$) is taken from the nRF905 transceiver datasheet as 10 and 50 kbps respectively. The number of bits used in the packets is that of 4 address bytes, 32 payload bytes and 2 CRC error correcting bytes. In this simulation, the summation of retransmissions according to noise was only added until the effect of the noise was smaller than 1% ($\delta^k < 0.01$).

$$t_{\text{preamble}} = \frac{\text{Amount of bits}}{BR} = \frac{10}{50000} = 200\mu s$$

$$t_{\text{postamble}} = 0$$

$$t_{\text{prop}} = \frac{\text{distance}}{c} = \frac{1000}{3 \times 10^8} = 3.3\mu s$$
\[ t_{\text{gen}} = 1ms \]
\[ t_b = \frac{1}{BR} = \frac{1}{50000} = 20us \]
\[ b_{\text{preq}} = b_{\text{prep}} = (4 + 32 + 2) \times 8 = 304 \text{ bits} \]
\[ t_{TTL} = 1.5 \times t_{\text{min}} \]

Figure 7.3: Average waiting time vs. number of stations using conventional RRP model

7.3.3 Queueing Approach

Round-Robin Polling can also be modeled using basic queueing theory. For more detail on queueing theory please refer to Section 7.2. The network under study has a finite source population and can be modeled as a queue with \(N\) sources, each with arrival rate \(\lambda\), and one server with service rate \(\mu\) (see Fig. 7.6). For stability, the total arrival rate can never exceed the service rate, otherwise the queue will grow boundlessly. New arrivals at each station arrive according to a Poisson distribution given by eq. 7.5. The sum of Poisson arrivals gives a distribution that is also Poisson, therefore the total arrival rate is equal to the sum of the individual arrival rates. Interarrival times of a Poisson distribution is exponentially distributed according to eq. 7.17. The arrival process is also memoryless, meaning that every new arrival is independent of other arrivals. This type of arrival is called a Markovian process. The base station is the only server in the network and follows a discrete distribution, seeing that service times are constant, which is classified as a general or deterministic distribution. The queueing model described thus
Figure 7.4: Average waiting time vs. number of stations for different topologies using conventional RRP

Figure 7.5: Average wait time vs. noise for different topologies using conventional RRP
far is that of a semi-Markovian queueing system. The equation derived in Section 7.2 for the total number of stations in the queue is therefore not valid, seeing that this is not a $M/M/1$ queue, but a $M/G/1$ queue. The waiting time in a $M/G/1$ queue can be determined using the Pollaczek-Khinchin formula given in eq. 7.18. The derivation of this formula can be found in [Hoc97].

\[ A(t) = \lambda e^{-\lambda t} \]  
\[ W = \frac{\rho \bar{x}}{2(1-\rho)} \left( 1 + C_b^2 \right) \]

With:

- $P_k(t)$ = Probability of $k$ arrivals during time $t$
- $\lambda$ = Event arrival rate
- $k$ = Number of arrivals
- $W$ = Average waiting time in queue
- $\rho$ = Channel utilization ($\rho = \lambda \bar{x} = \frac{\lambda}{\mu}$)
- $\bar{x}$ = Average service time (also $\bar{x} = \frac{1}{\mu}$)
- $C_b$ = Ratio between the standard deviation and mean ($C_b \equiv \frac{\sigma_x}{E[x]}$)

It must be remembered that with Round-Robin polling the base station is only communicating with one station at a time. A model of a RRP network with $N$ stations is shown in Fig. 7.6. The base station will scan through the whole network sequentially, starting by sending a poll request to station 1. The base station will now wait for the reply from that station, before sending a poll request to the next station. While the base station is scanning the other stations, new arrivals at station 1 cannot be serviced before all other stations have been serviced. It can be seen as if, when the station has been serviced, the server goes on vacation until this station is serviced again during the next scan cycle. The vacation starts once a station has successfully sent its data to the base station and only stops when it receives a new poll-request signal. Eq. 7.18 can now be expanded to eq. 7.19, as explained in [Hoc97]. In this equation $\overline{V}$ is the mean vacation.
time, while $\overline{V^2}$ is the second moment of the vacation distribution, which can be determined using the relationship stated in eq. 7.20.

$$W = \frac{\rho \bar{x}}{2(1-\rho)} (1 + C^2_b) + \frac{\overline{V^2}}{2\overline{V}}$$  \hspace{1cm} (7.19)$$

$$\overline{V^2} = \sigma^2_V + \overline{V^2}$$  \hspace{1cm} (7.20)$$

This formula can now be used to analyze the network at hand. The first value to determine is the service time. The service time is that time needed to service one station. The minimum service time needed to service 1 station is the same as for the conventional method, i.e. eq. 7.6. This equation, however, does not take noise and possible retransmissions into account. The same approach as with the conventional solution is used to add the noise effect. Assume $\delta$ is the fraction of packets that has to be retransmitted. Also remember that the mean value of a random process can be determined using eq. 7.21. The variance can then be determined using eq. 7.22. Both equation 7.21 and 7.22 were obtained from [Pee01]. The hat above the variables indicates that the calculated values are estimated values. Seeing that there is no closed form solution for this equation, 1000 test points will be taken and the mean and variance of that dataset will be taken. To simplify the derivation, retransmissions are limited to two times per packet. The number of packets transmitted successfully without retransmission is therefore $1000(1-\delta-\delta^2)$; those retransmitted once is $1000\delta$ and those retransmitted twice is $1000\delta^2$. Substituting all this into eq. 7.21 and 7.22 and also taking $\bar{x} \approx \overline{X}$ and $\sigma \approx \overline{\sigma}$, produces eq. 7.23 and 7.24. The reason for using 1000 data points is to ensure that each of the three possible states would have an integer number of points and, therefore, the sum of the three possible cases would be exactly 1000 points. These two equations can easily be solved using MATLAB. First create the 1000 element vector with the appropriate number of values for each of the three options. The mean and standard deviation can now be determined using the mean and std functions.

$$\overline{x_N} = \frac{1}{N} \sum_{n=1}^{N} x_n$$  \hspace{1cm} (7.21)$$

$$\overline{\sigma^2_X} = \frac{1}{N-1} \sum_{n=1}^{N} (X_n - \overline{X_N})^2$$  \hspace{1cm} (7.22)$$

$$\overline{x} = \frac{1}{1000} \left( \sum_{i=1}^{1000(1-\delta-\delta^2)} (t_{min}) + \sum_{i=1}^{1000\delta} (t_{min} + t_{TTL}) + \sum_{i=1}^{1000\delta^2} (t_{min} + 2t_{TTL}) \right)$$  \hspace{1cm} (7.23)$$

$$\overline{\sigma_x} = \sqrt{\frac{1}{999} \left( \sum_{i=1}^{1000(1-\delta-\delta^2)} (t_{min} - \overline{x})^2 + \sum_{i=1}^{1000\delta} (t_{min} + t_{TTL} - \overline{x})^2 + \sum_{i=1}^{1000\delta^2} (t_{min} + 2t_{TTL} - \overline{x})^2 \right)}$$  \hspace{1cm} (7.24)$$

Once again the number of stations is increased according to eq. 7.14 to compensate for the multi-hop effect. The vacant time is the total time needed to service all the other stations. The
vacant time and the standard vacant deviation can now be determined using eq. 7.25 and eq. 7.26 respectively.

\[
\bar{V} = (N_{\text{virtual}} - 1)(\bar{x}) \quad \text{(7.25)}
\]

\[
\sigma_V = (N_{\text{virtual}} - 1)(\sigma_x) \quad \text{(7.26)}
\]

The value of \( \lambda \) can now be chosen to determine the required values. The total time of a packet in the system can be determined by eq. 7.27. Applying Little’s theorem \((N = \lambda T)\) to the previous equation gives eq. 7.28, which is the total number of packets in the system. Eq. 7.29 gives the total number of packets waiting in the queue. All three these equations were obtained from [Hoc97]. It should be remembered that the number of packets in the queue may not exceed one, otherwise multiple transmissions would be needed to service each station. This is not the case with the model derived thus far and is therefore not allowed.

\[
T = \bar{x} + W \quad \text{(7.27)}
\]

\[
N = \rho + \frac{\rho^2}{2(1 - \rho)} \left(1 + C_b^2\right) + \frac{\lambda \bar{V}^2}{2V} \quad \text{(7.28)}
\]

\[
N_q = \lambda W = \frac{\rho^2}{2(1 - \rho)} \left(1 + C_b^2\right) + \frac{\lambda \bar{V}^2}{2V} \quad \text{(7.29)}
\]

The equations obtained for the RRP queueing model was again solved using MATLAB. The effect noise has on the RRP queue model is shown in Fig. 7.7 and 7.9. In the first figure the wait time is plotted against the number of stations. The RRP queue model reacts the same as the conventional RRP model (see Fig. 7.3), for an increase in the number of stations. A comparison between the two is shown in Fig. 7.8. Note how the latency of the queueing approach is always greater than that of the conventional approach by some constant. This value is that of \( \bar{x} \), which remains constant irrespective of the number of stations (see eq. 7.23).

A plot of the average waiting time vs. the arrival rate (\( \lambda \)) is shown in Fig. 7.9. When analyzing the results obtained in this figure, it should be remembered that the queue length may not exceed 1, seeing that the server will then not be able to service all stations fast enough. Taking this into account, it can be seen that the queue length grows faster for higher noise levels, therefore the queue reaches its maximum capacity faster than it does for the lower noise level graphs. With this type of network, the server can either service all stations fast enough, or it cannot. This explains the nature of the graph in Fig. 7.9. Once the arrival rate reaches a certain threshold point, the queue is no longer capable of storing more entities and the server can no longer service all the stations. The reason why the average waiting time does not increase for an increase in \( \lambda \), is because the vacation time (the time it takes to service all the other stations) is still the same. It should also be remembered that the arrival process is memoryless. This can be summarized by stating that the base station will always be able to service all stations in the network as long as the arrival rate is lower than the service rate.
Chapter 7 — Performance Predictions

### Figure 7.7: Effect of noise on the RRP queue model

![Graph showing the effect of noise on the RRP queue model.](image)

### Figure 7.8: Comparison between the conventional and the queue RRP model

![Graph comparing conventional RRP and queue RRP models with 10% noise and arrival rate of 0.1.](image)
Figure 7.9: Effect of noise and arrival rate on the RRP queue model

7.4 CSMA

7.4.1 Background

Contention protocols use strategies where the base station is not in full control as it is with RRP. Each station sends data whenever new entities arrive at that station. If no new entries arrive, no messages are sent and therefore the data traffic is reduced enormously. This may sound like the perfect solution, but there is a trade-off. With RRP, stations only transmit data when a poll-request signal is received from the base station and therefore collisions cannot occur. With contention protocols all stations can initiate data transfer at random intervals and, therefore, packets can collide if multiple stations transmit data at the same time. With this type of network, if the base station receives data from a station, it sends an ACK message to acknowledge that the packet has been received. As with RRP, a TTL timer is set, but this time at the station and not the base. If the station does not receive an ACK message after $t_{TTL}$, it retransmits the packet. It must be remembered that the ACK message is subject to the same noise levels as the original message and is therefore just as vulnerable to noise.

The protocol described thus far is called ALOHA. Clearly this form of communication would result in error whenever more than one station tries to transmit data at the same time. As described by [Wol02] the performance of this strategy deteriorates rapidly with an increase in arrival rate. To reduce the probability of multiple stations transmitting data at the same time,
the process of channel sniffing is introduced. Before transmitting data, a station first sniffs the communication channel to check whether another station is not already transmitting. If the channel is clear, the station would transmit its data, otherwise, according to some strategy, it would wait for the channel to become clear and then transmit its own data. This technique of sniffing can be used to reduce collisions, but it cannot completely remove their occurrence due to the hidden terminal effect, transmitter rise times and propagation times.

The hidden terminal effect can best be described at the hand of Fig. 7.10. If station C transmits data to the BS, station D would detect it, but station B would not be able to detect it. If station B wants to send data to the BS at the same time, it would sniff the communication channel and detect the channel as idle and would, therefore, send its own data as well and cause a collision. The same goes for D or B transmitting first. The hidden terminal effect has an exceptionally big influence on multi-hop. A level 3 station (A) would never be able to see the BS, otherwise itself would have been a level 2 station. Therefore, if station A tries to transmit data to B it would not detect the base station transmitting data to B, C or D and this would cause a collision if the BS sends data to any of its children, seeing that B would also receive this signal.

Before a transmitter can transmit data, it first needs some time ($t_{rise}$) to allow the transmitter to charge up. This time needed is referred to as rise time, which creates a vulnerable period during which a station would not detect that another station is already starting to transmit data. This vulnerable period is another source of collisions. The longer this vulnerable period, the higher the probability that transmissions from multiple stations would collide during this period. There is no way to reduce collisions caused by the hidden terminal effect, but techniques do exist to reduce rise time collisions.

One strategy used with CSMA, is that in which the channel is continuously sniffed. This is
called persistent CSMA (See: [KT75]), as the station persists in continually transmitting data. Persistent CSMA can be divided into two categories. The first is that of 1-persistent CSMA, where the station sniffs the channel continuously and then immediately (with probability of 1) transmits its own data once the channel goes idle. The reason for using this strategy is based on the fact that the channel is utilized to the maximum and not allowed to become idle. A disadvantage of this strategy is that there is a very high probability of collisions occurring due to rise times. If multiple stations were to generate new entries while another station is already transmitting data, all these stations would detect that the channel is busy and continue sniffing. Once the channel goes idle, all these stations would start transmitting at the same time, causing a collision. This collision probability can be reduced using p-persistent CSMA. With this strategy the channel is still sniffed continuously, but a station would only transmit new data with probability of $p$ once the channel goes idle. Another strategy implemented to reduce collisions is that of non-persistent CSMA. This strategy makes use of back-off periods. If a channel has data to send, it sniffs the communication channel. If the channel is idle, it would transmit its data, otherwise it would set a timer and go into back-off. Once the timer expires, the station would test the channel again and repeat the previous step. This back-off time is chosen randomly to try and reduce the probability of multiple stations going into back-off and then returning at the same time, which would increase the probability of collision due to rise times.

7.4.2 Collision probabilities

Clearly, as seen in Section 7.4.1, collisions have an enormous influence on CSMA network performance and should be studied more closely. In this section, the number of collisions caused by the hidden node effect and by rise times is analyzed statistically. The aim is to try and predict the number of errors that would occur due to these two factors.

7.4.2.1 Collisions due to the hidden terminal effect

In a multi-hop system every node would almost always have at least one hidden node. Looking at all the stations in Fig. 7.10 individually, it can be seen that for A, BS is hidden, for B, C and D are hidden, for C and D, B is hidden and for BS A is hidden. It can also be stated that any station on a level higher than two, would always have at least one hidden node that is 2 levels lower than itself. To try and determine the hidden terminal probability analytically, we use circles. Each circle represents the transmit radius of a station which is positioned at its centre. The assumptions made while deriving this model are:

- All stations have equal transmit radii.
- All stations are evenly distributed within the receptive area (no clustering).
Chapter 7 — Performance Predictions

Figure 7.11: Two circles intersecting: Worst case scenario

- The signal strength is equal throughout the transmit radius.

First we start by looking at two circles, as shown in Fig. 7.11. This figure gives the worst case scenario where two stations (situated at point A and B) are just within transmitting range of each other. We start the analysis by determining the intersection points C and D. This can be done by setting up the equations for the two circles (eq. 7.30 and 7.31) and then solving for the two intersection points using eq. 7.32 to 7.34 (See: [Wil]). After determining the intersection points, the common intersection area can be determined using eq. 7.35 as explained in [Ant07]. The common intersection area is found to be 39.1%. This area is visible to both stations and any stations within this region would always be the lowest level possible. The area not visible to both stations is 60.9%, as determined by eq. 7.36. If station A is a level 1 station and B is a level 2 station, then level 3 stations can only be found in this 60.9% of station B’s receptive area to the right of station B. It should be remembered that this is the biggest available area, taking into account that if the stations are closer to each other, the common intersect area would increase, causing the available area to decrease. Seeing that this is an approximation, this value used is 60% to simplify calculations.

\[
(x^2 - x_1^2) + (y^2 - y_1^2) = r_1^2
\]  
(7.30)

\[
(x^2 - x_2^2) + (y^2 - y_2^2) = r_2^2
\]  
(7.31)

\[
x = \frac{x_2 + x_1}{2} + \frac{(x_2 - x_1)(r_1^2 - r_2^2)}{2d^2} \pm \frac{y_2 - y_1}{2d^2} \sqrt{\left(r_1 + r_2\right)^2 - d^2} \left(d^2 - (r_2 - r_1)^2\right)
\]  
(7.32)

\[
y = \frac{y_2 + y_1}{2} + \frac{(y_2 - y_1)(r_1^2 - r_2^2)}{2d^2} \pm \frac{x_2 - x_1}{2d^2} \sqrt{\left(r_1 + r_2\right)^2 - d^2} \left(d^2 - (r_2 - r_1)^2\right)
\]  
(7.33)

\[
d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}
\]  
(7.34)

\[
Area = r^2 \left(q - \sin q\right) \text{ with } q = 2 \arccos \frac{d}{2r}
\]  
(7.35)
\[ \text{Area}_{\text{not visible}} = 1 - \frac{\text{Area}}{\text{total area of circle}} = 1 - \frac{r^2 (q - \sin q)}{2\pi r^2} \] (7.36)

From the values derived thus far, it is apparent that only 60% of a station’s transmit radius can be used to acquire new nodes, given the fact that the other 40% is also visible to the lower level station, which would acquire any stations in that region for itself. The probability under consideration is one where another station is added to the graph above, to obtain the graph shown in Fig. 7.12. This figure shows the worst case scenario if there are more level 3 stations than level 2 stations. The common area shared amongst the level 2 and level 3 station would be at a minimum if the level 3 station is just within the level 2 station’s radius, but just outside the level 1 station’s radius, closest to the intersection point of the level 1 and level 2 circles. In this figure, the shared area between the level 1 and level 2 stations is the same as that discussed in the two circle scenario and is given by the sum of areas A and C. Areas A and B give the shared area between circles 2 and 3.

Figure 7.12: Three circles intersect: Worst case scenario for more level 3 stations than level 2 stations

To further analyze this scenario the region shared between all 3 circles has to be determined. This is done using the derivation presented by [Few06]. With this derivation the origin of the coordinate system are placed at the centre of the level 1 station, while the centre of the level 2 station is placed on the \(x\) axis. The \(y\) coordinate of the level 3 station is chosen to be positive. Figure 7.13 shows the coordinate system used by Fewell. Note the addition of the \(x'\) and \(x''\) systems to simplify the calculation of the intersecting points of the third circle. Also note that the three intersecting points \((x_{12}, y_{12})\), \((x_{23}, y_{23})\) and \((x_{13}, y_{13})\) forms a triangle within the circles, hence the name circular triangle.

With his analysis the following seven steps must be followed.

- Step 1. Test whether circles 1 and 2 intersect by using eq. 7.37, with \(r_k\) the respective radii and \(d_{jk}\) the distance between the two points. If this is not satisfied, the algorithm
cannot be completed.

\[ r_1 - r_2 < d_{12} < r_1 + r_2 \] (7.37)

- Step 2. Calculate the relevant intersection points. using eq. 7.38.

\[ x_{12} = \frac{r_1^2 - r_2^2 + d_{12}^2}{2d_{12}}, \quad y_{12} = \frac{1}{d_{12} \sqrt{2d_{12}^2(r_1^2 + r_2^2) - (r_1^2 - r_2^2)^2 - d_{12}^4}} \] (7.38)

- Step 3. Calculate the values of the sines and cosines of the angles \( \theta' \) and \( \theta'' \).

\[ \cos \theta' = \frac{d_{12}^2 + d_{13}^2 - d_{23}^2}{2d_{12}d_{23}}, \quad \sin \theta' = \sqrt{1 - \cos^2 \theta'} \] (7.39)

\[ \cos \theta'' = \frac{d_{12}^2 + d_{23}^2 - d_{13}^2}{2d_{12}d_{23}}, \quad \sin \theta'' = \sqrt{1 - \cos^2 \theta''} \] (7.40)

- Step 4. Verify that circle 3 is placed so as to form a circular triangle. If the equations below are not satisfied, the calculations cannot be completed.

\[ (x_{12} - d_{13}\cos \theta')^2 + (y_{12} - d_{13}\sin \theta')^2 < r_3^2 \] (7.41)

\[ (x_{12} - d_{13}\cos \theta')^2 + (y_{12} + d_{13}\sin \theta')^2 > r_3^2 \] (7.42)

- Step 5. Calculate the intersection points caused by circle 3.

\[ x_{13}' = \frac{r_1^2 - r_3^2 + d_{13}^2}{2d_{13}}, \quad y_{13}' = -\frac{1}{2d_{13}} \sqrt{2d_{13}^2(r_1^2 + r_3^2) - (r_1^2 - r_3^2)^2 - d_{13}^4} \] (7.43)

\[ x_{13} = x_{13}' \cos \theta' - y_{13}' \sin \theta', \quad y_{13} = x_{13}' \sin \theta' + y_{13}' \cos \theta' \] (7.44)

\[ x_{23}'' = \frac{r_2^2 - r_3^2 + d_{23}^2}{2d_{23}}, \quad y_{23}'' = \frac{1}{2d_{23}} \sqrt{2d_{23}^2(r_2^2 + r_3^2) - (r_2^2 - r_3^2)^2 - d_{23}^4} \] (7.45)

\[ x_{23} = x_{23}'' \cos \theta'' - y_{23}'' \sin \theta'' + d_{12}, \quad y_{23} = x_{23}'' \sin \theta'' + y_{23}'' \cos \theta'' \] (7.46)
• Step 6. Calculate the intersection chord lengths.

\[ c_k^2 = (x_{ik} - x_{jk})^2 + (y_{ik} - y_{jk})^2 \]  

(7.47)

• Step 7. Determine whether more than half of circle 3 is included in the circular triangle, so as to choose the correct expression for the area.

\[ d_{13} \sin \theta' < y_{13} + \frac{y_{23} - y_{13}}{x_{23} - x_{13}} (d_{13} \cos \theta' - x_{13}) \]  

(7.48)

\[ A = \frac{1}{4} \sqrt{(c_1 + c_2 + c_3)(c_2 + c_3 - c_1)(c_1 + c_3 - c_2)(c_1 + c_2 - c_3) + \sum_{k=1}^{3} \left( \frac{r_k^2 \arcsin \frac{c_k}{2r_k}}{4} \right) \left\{ \begin{array}{ll}
\frac{c_k}{4} \sqrt{4r_3^2 - c_3^2} & \text{(Eq. 7.48 true)} \\
\frac{c_k}{4} \sqrt{4r_3^2 - c_3^2} & \text{(Eq. 7.48 false)}
\end{array} \right.} \]  

(7.49)

These seven steps were fed into MATLAB and the area was calculated as 22.43% of the total area of one circle. As discussed, the common area between two nodes is 40%, therefore the approximate area of section B and C can now be derived as 17.5% of the total area. This portion (B) however is equal to 29% \((\frac{17.5}{60} \times 100)\) of the level 2 station’s available area. Keeping in mind that all stations are assumed to be equally spaced and positioned within this 60% region, and assuming the level 2 station has \(N\) level 3 stations, then the level 3 station would only be able to see those within region B. The number of hidden level 3 nodes is, therefore, 71% of the stations (excluding the node under study). The grandparent node must always be added, seeing that this station would never be visible. Another term that must be added is that of the other level 2 stations that transmit within the same frequency band and are within earshot of this level 2 station. These level 2 stations would generate noise at the level 2 station under consideration, and would cause errors if the level 3 station were to transmit at the same time. The stations described above are those within the 40% shared region (A and C). Region A would also be visible to the level three station, and therefore only region C is a problem. This area is 17.5% of the level 2 stations. It must be remembered that the level 2 stations are spaced evenly throughout the whole transmit radius of the level 1 station, causing 17.5% of the level 2 stations also to be hidden. The worst case statistical number of hidden nodes for a level 3 station is, therefore, given by eq. 7.50.

\[ N_{hidden\ ac1} = 0.71(N_{level3} - 1) + 0.18(N_{level2} - 1) + 1 \]  

(7.50)

For a level higher than three (take K) the same derivation is used for the first term. For the second term, however, it must be remembered that all level K-1 stations would only be present in 60% of the level K-2 station’s area. The number of level K-1 nodes within earshot of the level K-1 node under study, would therefore be 66.7% \((\frac{66}{60} \times 100)\) of the level K-1 nodes. Taking into account that the level K-1 stations in region A would also be seen by the level K station, only those within region C are a problem. That is only 29.2% \((\frac{17.5}{60} \times 66.7)\) of the level K-1 stations. For a station with level higher than 3, the number of hidden nodes is given by eq. 7.51. It must be stated that this analysis is only for transmission downwards (towards the base) and does
not take nodes hidden due to upward transmission into account. The probability of collisions upwards is the same as that case stated in eq. 7.51, replacing $N_{\text{level } K-1}$ with $N_{\text{level } K+1}$. The reason for using only one direction is that the channel would only be transmitting in one direction at a time. Transmission in opposite directions would not cause errors for the receiving station, but could cause errors for the other stations which might be receiving messages from still further stations. A CSMA network is modeled as a birth-death-process (described in Section 7.4.3.2), which limits the network to changing to only one neighbouring state at a time. This limitation prevents multiple transitions, which implies that only one station can transmit data at a time, but the network is still subject to multiple collisions.

$$N_{\text{hidden } K \text{ sc1}} = 0.71(N_{\text{level } K-1}) + 0.29(N_{\text{level } (K-1) - 1}) + 1 \quad (7.51)$$

If there are more level 2 stations than level 3 stations, the worst case scenario would be where the level 3 node is on the edge of the level 2 station’s range, without crossing the level 1 station radius as shown in Fig. 7.14. With this case the level 3 station would not see 33% ($\left(1 - \frac{40}{60}\right) \times 100$) of the other level 3 stations, which are represented by regions C and D. The level 3 station would also miss 40% of the level 2 stations (region A) which are within transmit radius of the level 2 station under study. The number of hidden nodes for this figure is therefore given by eq. 7.52. If this figure is expanded to a system with the highest level greater than 3, the last term would change to 67% ($\left(\frac{40}{60}\right) \times 100$) which would result in eq. 7.53.

$$N_{\text{hidden sc2}} = 0.33(N_{\text{level 3}} - 1) + 0.4(N_{\text{level 2}} - 1) + 1 \quad (7.52)$$

$$N_{\text{hidden } K \text{ sc2}} = 0.33(N_{\text{level } K-1}) + 0.67(N_{\text{level } (K-1) - 1}) + 1 \quad (7.53)$$

The best case would be where the level 3 station is as close as possible to the level 2 station, but just out of reach of the level 1 station. In this case it would be able to see all other level 3 stations, as well as all other level 2 stations. There would, therefore, only be one hidden node,
the level 1 node itself.

\[ N_{\text{hidden sc3}} = 1 \]  

(7.54)

Seeing that the network under consideration is mostly dependent on multi-hop, the 3 limiting multi-hop equations are given by eq. 7.51, 7.53 and 7.54. As the stations are randomly distributed and the probability of all the different positions discussed above is equal, the best estimation would be to take the average of the three limiting scenarios. If the average of these equations is taken the result in eq. 7.55 is obtained, where \( k \) is the level under investigation.

\[ N_{\text{hidden avg k}} = \frac{N_{\text{hidden K sc1}} + N_{\text{hidden K sc2}} + N_{\text{hidden sc3}}}{3} \]  

(7.55)

\[ = 0.35(N_{\text{level k}} - 1) + 0.32(N_{\text{level (k-1)}} - 1) + 1 \]

Another factor that might influence the hidden node effect is the terrain. We add a variable \( \delta_{\text{terrain}} \), which is a fraction of how many nodes would be hidden from each other due to the terrain. Remember that the results obtained earlier are just taking the amount of hidden nodes due to the network topology into account. If this \( \delta_{\text{terrain}} \) variable is added, the number of visible nodes can be scaled by this factor to include obstacles. Remember that only the visible nodes can be affected by this factor. Eq. 7.55 can now be extended using this factor to obtain the result in eq. 7.56, which is a measure of how many nodes would be hidden to the station. With the number of hidden nodes having been studied, the probability of a hidden node can now be determined using eq. 7.57.

\[ N_{\text{avg k}} = N_{\text{hidden avg k}} + \delta_{\text{terrain}}(N_{\text{level k}} + N_{\text{level (k-1)}} - 2 - N_{\text{hidden avg k}}) \]  

(7.56)

\[ \delta_{\text{hid}} = \frac{N_{\text{avg k}}}{N_{\text{level k}} + N_{\text{level (k-1)}}} \]  

(7.57)

The previous equation shows that \( \delta_{\text{hid}} \) of the stations, would be hidden. An error would only occur if the communication channel is busy and if channels were to transmit data simultaneously. As explained in Section 7.4.3, the total number of stations in the network is given by \( N \). The number of stations that do not currently have data to send is given by \( n \), and only these stations can generate new data with individual arrival rates of \( \lambda \) per station. Stations which already have data \((N - n)\), can only add new arrivals if they come out of back-off at a rate of \( \gamma \). Taking all this into account leads to eq. 7.58, which gives the total arrival rate due to hidden terminals. Each new message provokes an ACK message and therefore the rate is doubled, as shown. Collisions can only occur if one or more stations transmit data while another station is already transmitting. Collisions would therefore occur only if a station tries to transmit data while the channel is already in use. The error arrival rate caused by hidden terminals can, therefore, be determined using eq. 7.59. The fraction of the messages that will have to be retransmitted can now be determined by dividing the error rate by the total arrival rate, as shown in eq. 7.60.

\[ \lambda_{\text{hidden}} = 2\delta_{\text{hid}}[n\lambda + (N - n)\gamma] \]  

(7.58)

\[ \lambda_{\text{error hid}} = (\lambda_{\text{hidden}})(\rho) \]  

(7.59)

\[ \delta_{\text{coll hid}} = \frac{\lambda_{\text{error hid}}}{\lambda_{\text{total}}} = \frac{2\rho\delta_{\text{hid}}[n\lambda + (N - n)\gamma]}{n\lambda + (N - n)\gamma} = 2\rho\delta_{\text{hid}} \]  

(7.60)
Note that this case studied thus far has been with occurrence of multi-hops. In the case where no multi-hop occurs, the derivation can be reduced enormously. In the case of no multi-hops, the worst case is where a station is on the edge of the server’s transmission radius and can therefore see only 40% of the server’s transmission area. If we assume that all stations are equally spaced, the hidden node effect would at worst be \((0.6 \times (N - 1))\) and would be 0 in the best case. An average should be taken, but it should also be doubled to take ACK messages into account. The collision probability would reduce to that of eq. 7.61, where \(\delta_{\text{coll}} = 0.3\).

\[
\delta_{\text{coll hid nm}} = 0.6 \rho \quad (7.61)
\]

### 7.4.2.2 Collisions due to rise times

Before a station transmits data, it sniffs the communication channel to see if the channel is busy or idle. If it is idle it transmits its data, otherwise it goes into back-off. However, there is a problem. Transmission does not occur instantaneously because each transmitter needs a period of time \((t_{\text{rise}})\) to charge up before transmission can begin. A very short time is taken for the signal to propagate from the station to the next station, that of \(\tau\). These two factors create a vulnerable period in which errors can occur, even at stations that are not hidden from each other. This vulnerable period for one message is given by eq. 7.62.

\[
t_v = t_{\text{rise}} + \tau, \quad \text{with} \quad \tau = \frac{d}{c} \quad (7.62)
\]

A collision due to this vulnerable period would only occur if more than 1 station tries to transmit data during this vulnerable time. For an error to occur, the stations have to start transmitting within this initial \(t_v\) seconds, when the channel is utilized. It is obvious that the number of collisions is dependent on the utilization of the channel, as well as the arrival rate of new messages and messages returning from back-off. The mean arrival rate is given by eq. 7.63. The multiplication by 2, incorporates ACK messages generated automatically on reception of a message. The factor \((1 - \delta_{\text{hid}})\) is used to exclude stations that are not visible to this station, because the errors caused by these stations are already accounted for by the hidden terminal effect. It should be restated that we are only interested in the mean number of arrivals that would occur in \(t_v\) seconds of the utilized time. The error arrival rate due to vulnerable time collisions can now be determined using eq. 7.64. The probability can now be derived by dividing the error rate by the total arrival rate as shown in eq. 7.65. For the no multi-hop case this equation reduces to that of eq. 7.66.

\[
\lambda_{tv} = 2(1 - \delta_{\text{hid}}) [n\lambda + (N - n)\gamma] \quad (7.63)
\]

\[
\lambda_{\text{error tv}} = (\lambda_{tv})(t_v)(\rho) \quad (7.64)
\]

\[
\frac{\lambda_{\text{error tv}}}{\lambda_{\text{total}}} = 2\rho t_v (1 - \delta_{\text{hid}}) \quad (7.65)
\]

\[
\delta_{\text{coll tv nm}} = 1.4 \rho t_v \quad (7.66)
\]
7.4.3 Setting up the Model

The protocol used in this study as a point of departure, is non-persistent CSMA as described in Section 7.4.1. As stated earlier in Section 7.2.2, every queueing system can be characterized by six basic parameters. The network under study can be described as follow:

- **Source population**
  The chosen network can be modeled by an infinite source population queue in tandem with a finite source population queue as discussed in [Nic06]. This model is shown in Fig. 7.15. The infinite source population represents the number of possible events generated at each station (system A). The number of stations in the network is, however, limited and is modeled by a finite source population queue (system B). These two queues are internally linked, but have to be solved separately.

- **PDF of the arrival process**
  Arrivals occur completely at random and are modeled using the Poisson distribution, given by eq. 7.5. The Poisson distribution is often used to model completely random events. The new arrivals enter each station’s individual queue at constant rate $\lambda$. If the station’s queue is empty the station will try to transmit this newly arrived packet. If the queue is not empty, the new data would be added to the older data and both would still be transmitted as one packet. It is assumed that the system is stable and, therefore, that all stations would be serviced. Even though some messages may stay in a station’s queue longer than others, the total arrival rate at the base station remains constant. This happens because all messages are serviced over a long period of time, independent of the time they have spent in the queues.

- **Number of servers**
  Even though each station acts as a server to all stations in levels higher than itself, there still is effectively only one server. All data has to be logged by the base station and, therefore, all messages are eventually queued up in its buffer. This system therefore only has one server.

- **PDF of the service process**
  As seen above, the system only has one server and we should therefore focus only on that server. The PDF of this server is difficult to motivate. The hardware forces the use of constant packet lengths, which would generate a service PDF close to that of a deterministic process. The deterministic process would be very difficult to analyze and it would be much more convenient if this process were Markovian. Let us assume the process is Markovian. To justify this, it is restated that both the arrival process and additive noise effects are truly random and therefore have a random influence on the service process. It is also stated that the processing time at the base station would differ depending on the
kind of message received. Some messages have to be retransmitted while others don’t, which also provides some sort of unpredictability. The back-off time used for the different packages is also completely random. All these factors together are used as motivation for modeling using a Markovian process with Poisson distribution.

- **Queueing discipline**
  The FIFO queueing discipline is used to reduce the number of retransmissions caused by TTL timers.

- **Queue buffer space**
  The buffer size is assumed to be equal to that of the total number of stations in the network. No packages would therefore be lost due to buffer overflow or blocking.

![Figure 7.15: CSMA Jackson network model](image)

The queue described above is classified as an $M/M/1$ queue and therefore Jackson’s theorems can be applied. Jackson’s theorems allow for the decomposition of complex networks into smaller sub-networks. The network described above consists of two major sub-networks.

### 7.4.3.1 Modeling the infinite queue

As proposed by [Nic06], the mean service time of the first queue is taken to be equal to the mean time a message spends in the second queue. If $T_A$ is the time spent in the first queue and $T_B$ is the time spent in the second queue, then the service rate of the first queue (system A) is given by eq. 7.67.

$$\mu_A = \frac{1}{T_B} \quad (7.67)$$

The time the packet spends in the first queue can now be found using eq. 7.3. This equation is restated in eq. 7.68 for convenience. From the derivation thus far it is apparent that each
station services all events generated at that station. The rate at which events leave this system to queue up in the second system is, therefore, exactly the same as the input rate. New events are therefore generated at the second queue at rate $\lambda$ per station.

$$T_A = \frac{1}{\mu_A - \lambda}$$ (7.68)

### 7.4.3.2 Modeling the finite queue using a state transition matrix

The number of stations in the network is limited and is modeled by a finite source population queue (system B). With this type of queue the same approach as was used with the infinite source type cannot be used, because the arrival rate is no longer constant, but a function of the number of stations that do not currently have a message. With finite source queues every station is currently either trying to transmit data or it is idling. The total number of stations is given by $N$, while the number of idle stations is given by $n$. All these independent station networks can be combined using Jackson’s theorems. According to Jackson, a combination of a finite number of independent Poisson distributed data streams, would produce another Poisson stream with a mean equal to the sum of the individual means ([RW06]). The arrival rate at the second queue is therefore $n\lambda$. A closed form mathematical solution rarely exists for an exact finite source case and therefore an alternative solution has to be found.

The finite queue is solved using a state transition matrix as explained in [Wol02]. This analysis assumes that a system is in equilibrium and therefore the number of stations in the system is constant during a certain time period. Each equilibrium state is represented individually and they are placed next to each other, as can be seen in Fig. 7.16. If new events occur, this would cause the system to move from one state to a neighbouring state. Only one transition can occur at a time. This process is referred to as a birth-death process and is a special case of the Markov chain, where each event can initiate transitions only from one neighbouring state to another ([Hoc97]). The system must always be in a certain state and therefore a state probability exists for all states. This probability is given by $P(n,m)$ where $n$ denotes the number of idle stations in the network and $m$ shows whether the server is busy or not.

![Figure 7.16: Basic state transition model](image-url)
The arrival and service rates between neighbouring states in a system with \( N \) nodes are shown in Fig. 7.16. In the initial state \( P(N,0) \), no station has data and the server is free. In this state all stations can send data and the rate at which new messages can leave this state to move to \( P(N - 1,1) \), is therefore \( N\lambda \). From this state there are two ways out. If a new arrival occurs (with arrival rate \((N - 1)\lambda\)) the station moves to \( P(N - 2,1) \) which is also in “busy” state, while if a message is serviced, it moves back to \( P(N,0) \) at rate \( \mu \). If a message is serviced while the current state is “busy” and has more than 1 station requesting service, then the state has to revert to an idle state, from where the other stations requesting service can come out of back-off, or new messages can arrive at rate \( k\gamma \) and \((N - k)\lambda \) respectively (\( k \) is the number of stations waiting to be serviced). The rate at which back-off re-entry occurs, is dependent on the number of stations still waiting to be serviced. A prerequisite of the back-off state is that at least one station has to be waiting to be serviced. It is therefore not possible to be in back-off state when all stations are idle. It is, however, possible to be in back-off state when no station is idling. This would happen if the last idle station arrives with a new message before the old messages come out of back-off.

For this system, all transition rates are known. The system is also assumed to be in equilibrium. Balance equations can now be written for all the individual states. These balance equations can easily be set up directly from the figure, by taking all data flow out of a station as positive. The balance equation for the first state is shown in eq. 7.69, while the equation for state \( P(N - k,1) \) is shown in eq. 7.70. The same approach is used to write all the other equations. The last equation needed is that shown in eq. 7.71, which states that the sum of the probabilities is 1.

\[
N\lambda P(N,0) - \mu P(N - 1,1) = 0 \quad (7.69)
\]

\[
[(N-k)\lambda + \mu]P(N-k,1) - (N-k+1)\lambda[P(N-k+1,1) + P(N-k+1,0)] - k\gamma P(N-k,0) = 0 \quad (7.70)
\]

\[
\sum P(n,m) = 1 \quad (7.71)
\]

All these equations are now converted to a matrix equation which is in the form shown in eq. 7.72. In this equation, \( A \) contains all the transition rates, \( B \) contains all the probabilities and \( C \) is a zero vector. If the last equation (that of eq. 7.71) is added to the matrix equation, it adds a row containing 1s to the \( A \) matrix and a 1 to the \( C \) vector. For \( N \) stations, the \( A \) matrix would have dimensions \( 2N + 1 \times 2N \), the \( B \) matrix’s dimensions would be \( 2N \times 1 \) and that of the \( C \) matrix would be \( 2N + 1 \times 1 \). The \( A \) matrix is referred to as the state transition matrix. To be able to solve the probabilities, the inverse of matrix \( A \) has to be taken to convert eq. 7.72 to eq. 7.73. \( A \), however, is not a square matrix and therefore the inverse does not exist. To solve this the pseudo inverse (also known as Moore-Penrose inverse) has to be determined.

\[
AB = C \quad (7.72)
\]

\[
B = A^{-1}C \quad (7.73)
\]

The pseudo inverse (as defined in eq. 7.74) is defined and unique for all matrices whose entries are real or complex numbers and is denoted as \( A^+ \). As described in [Pse] and [Lay03], the
pseudo inverse can be determined using singular value decomposition (SVD). The matrix under study is a special case where all columns are independent \((\text{rank} = 2N)\), which reduces eq. 7.74 to eq. 7.75. This is clearly the case for the matrix under consideration if either \(\lambda\) or \(\mu\) is greater than 0. The value of \(B\) can now be determined using eq. 7.76. This matrix can easily be solved using the left division operator in MATLAB \((B = A \backslash C)\). When using the left division operator MATLAB uses the method of least squares to determine the values. This is an approximation, but gives very good results.

\[
A^+ = \lim_{\delta \to 0} A^T A + \delta I = 1 \quad (7.74)
\]

\[
A^+ = (A^T A)^{-1} A T \quad (7.75)
\]

\[
B = (A^T A)^{-1} A^T C \quad (7.76)
\]

With the probabilities known, the average queue length can now be determined. It should be noted that the queue length also includes the packets being serviced by the server. The queue length for each state is, therefore, the same as the number of stations requesting service in that state. The total queue length can now be determined by summing the number of stations requesting service in each state, and multiplying by the probability that the system is in that state. The average system queue length is determined using eq. 7.77.

\[
L = \sum_{i=0}^{N-1} P(i,1) \times (N-i) + \sum_{j=1}^{N} P(j,0) \times (N-j) \quad (7.77)
\]

As stated earlier, the arrival rate for the finite source case is \(n\lambda\), where \(n\) gives the number of idle stations. With eq. 7.77 the average number of stations in the queueing system is calculated. The average arrival rate can now be rewritten as \((N-L)\lambda\). The average time spent in system \(B\), according to Little's law, can now be calculated by eq. 7.78. The total time in the network is given by eq. 7.79.

\[
T_B = \frac{L}{\lambda_B} = \frac{L}{(N-L)\lambda} \quad (7.78)
\]

\[
T_{\text{network}} = T_A + T_B = \frac{1}{\mu_A - \lambda} + \frac{L}{(N-L)\lambda}, \quad \text{with} \quad \mu_A = \frac{1}{T_B} = \frac{(N-L)\lambda}{L} \quad (7.79)
\]

An example of the matrix equation \((AB = C)\) for a system with 4 stations \((N = 4)\), is shown below. With this model the arrival rate \((\lambda)\) and the back-off rate \((\gamma)\) can be chosen randomly. This model would then predict the response to the settings chosen. The service rate \((\mu_B)\) can not be chosen, and has to be determined analytically. The time the server needs to service one station is simply the average cycle time needed to transmit one packet. This system only has one server and thus the service time is the same as in the RRP case \((t_{\text{min}}\text{ shown in eq. 7.6})\). It should be noted that with CSMA, a DATA packet and an ACK packet are sent, in contrast to the poll-reply and poll-request packets. This does not have an influence, as the packet sizes are
the same. The service rate is therefore given by eq. 7.80.

\[
\begin{bmatrix}
4\lambda & -\mu_4 & 0 & 0 & 0 & 0 & 0 & 0 \\
-4\lambda & 3\lambda + \mu_4 & 0 & 0 & 0 & -\gamma & 0 & 0 \\
0 & -3\lambda & 2\lambda + \mu_3 & 0 & 0 & -3\lambda & -2\gamma & 0 \\
0 & 0 & -2\lambda & \lambda + \mu_2 & 0 & 0 & -2\lambda & -3\gamma \\
0 & 0 & 0 & -\lambda & \mu_1 & 0 & 0 & -\lambda \\
0 & 0 & 0 & -\mu_3 & 0 & 0 & 3\lambda + \gamma & 0 & 0 \\
0 & 0 & 0 & -\mu_2 & 0 & 0 & 2\lambda + 2\gamma & 0 \\
0 & 0 & 0 & 0 & -\mu_1 & 0 & 0 & \lambda + 3\gamma \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
P(4,0) \\
P(3,1) \\
P(2,1) \\
P(1,1) \\
P(0,1) \\
P(3,0) \\
P(2,0) \\
P(1,0)
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
1
\end{bmatrix}
\]

\[\mu_{\text{no noise}} = \frac{1}{t_{\text{min}}}\] (7.80)

This model does not take retransmissions into account. As discussed in [Wol02], the effect of burst noise is added by extending the service time. This is done by assuming that a certain fraction (\(\delta_{\text{burst}}\)) of the packets will have to be retransmitted. The service rate is now changed to that of eq. 7.81, which includes the burst noise effect. This model also does not take multi-hop into account. The approach employed with RRP, that of virtual stations, cannot be used with the CSMA model. For the RRP case this assumption is valid, because every station has to wait for all other stations to be serviced before it can be serviced again. The CSMA model is interrupt driven and therefore, the latency of a packet is dependent on the number of hops. With CSMA, a station would be serviced immediately the communication channel goes idle. If the virtual station approach used with RRP were to be used for CSMA, the latency for a multi-hop station would be the same as for a no-hop station if the arrival rate is low. As explained by [Nic06], the effect of multi-hops could be added by extending the service time by the mean number of hops. The effective service time can now be derived using eq. 7.82.

\[\mu_{\text{eff}} = \frac{1}{T_{\text{eff}}} = \frac{1}{t_{\text{min}} + (\delta_{\text{burst}})(t_{\text{TTL}})}\] (7.81)

\[\mu_{\text{eff2}} = \frac{1}{T_{\text{eff2}}} = \frac{1}{h_{\text{mean}} \times (t_{\text{min}} + (\delta_{\text{burst}})(t_{\text{TTL}}))}\] (7.82)

The model derived thus far was solved using MATLAB, and the results obtained are shown in fig. 7.17 through to 7.21. For these calculations the TTL timer is set to twice the minimum time needed to send one message (\(t_{\text{TTL}} = 2t_{\text{min}}\)). Fig. 7.17 shows the effect that burst noise has on a network with 10 stations. The system utilization for this network is shown in fig. 7.18. Note the increase in utilization due to packet retransmission. The model derived by [Nic06] does not take utilization increase into account. This increase in utilization is very important, because the level of hidden terminal noise is dependent on the utilization factor as described in Section 7.4.2.1. Note that it is assumed that any station would only be able to retransmit a packet once. If multiple retransmissions are desired, the burst noise level can be increased accordingly. The total system latency consists of two individual latencies, that of system A (\(T_A\)) and system B (\(T_B\)) respectively. The time spent in each of these systems is shown in Fig. 7.19. The total
queue length, as well as the individual queue lengths for the different noise levels, is shown in Fig. 7.20. The last figure, shows the effect that multi-hops has on the CSMA network.

![Figure 7.17: Effect of noise on CSMA](image)

---
Chapter 7 — Performance Predictions

Figure 7.18: Utilization for CSMA model

Figure 7.19: Time spent in each queue vs different noise levels
Figure 7.20: Queue lengths for different noise levels
7.4.4 Extending the model to include collisions

The model described in the previous section does not take collisions into account. The probability that collisions would occur, are discussed in Section 7.4.2. These probabilities can now be added to the previous model to acquire a more realistic result. The probabilities can either be used to increase the arrival rate due to noise, or can be used to decrease the service rate. The latter method is already used to model the burst noise and is, therefore, only expanded to allow for collisions. Eq. 7.81 is now expanded to eq. 7.83, which also includes collision errors. Equations 7.61 and 7.66 are used to represent the collision probabilities and are restated in eq. 7.85 and 7.84 for convenience. The collision errors used differ from those used in [Wol02] and [Nic06].

The model used by [Nic06] uses the same approach as [Wol02], but tries to expand the model to incorporate new arrivals. His approach models the noise as $n\lambda 2t_r\gamma (1 - \rho) + 2\tau$. It should be noted that the combined arrival rate is given by the sum of the arrival rates and not by the product, as proposed by [Nic06]. This approach is not valid. Also note that the added $2\tau$ in both of these approaches represents the propagation delay incurred in sending one message. With the model proposed in this thesis, this value is embedded into $t_{\text{min}}$.

$$\mu_{\text{coll}} = \frac{1}{T_{\text{eff coll}}} = \frac{1}{h_{\text{mean}} \left[ t_{\text{min}} + (\delta_{\text{burst}} + \delta_{\text{coll hid}} + \delta_{\text{coll tv}})/(t_{\text{TTL}}) \right]}$$ (7.83)
These additions were added to the MATLAB model used in the previous section, and the results obtained are shown in Fig. 7.22 through to 7.28. For all these simulations the TTL timer was set to $t_{TTL} = 2t_{min}$, the back-off timer set to 200 ms and 10 stations were used. The effect hidden nodes have on the latency is shown in Fig. 7.22. Note the dramatic increase in latency when the arrival rate gets high. Note especially how the latency increases according to the utilization, shown in Fig. 7.23. The increase in utilization is caused by retransmissions, which generates additional arrivals. The increase in queue lengths and, therefore, time spent in each queue is also shown in Figs. 7.24 and 7.25 respectively. With the system under study, the rise time needed by each transceiver is very short. This greatly decreases the probability of collisions caused by these rise times. The effect rise times have on the latency is shown in Fig. 7.26 for a network with rise times of 6 ms. Note the increase from the normal value of 650us. The effect that both burst noise and collisions have on a typical CSMA network have been analyzed thus far. They are now combined and shown in Fig. 7.27. Note especially the effect $\lambda$ has on the different noise probabilities, shown in Fig. 7.28. The rise time collisions for this network with $t_{rise} = 650us$ is negligible, while the hidden node probabilities grow enormously with increase in arrivals. It should also be noted that an increase in new arrivals would also cause an increase in back-off arrivals.
Chapter 7 — Performance Predictions

Figure 7.23: Effect of hidden nodes on utilization for CSMA networks

Figure 7.24: Effect of hidden terminals on queue lengths for CSMA networks
Figure 7.25: Effect of hidden nodes on time spent in each queue for CSMA networks

Figure 7.26: Effect of rise time on latency for CSMA networks
Figure 7.27: Effect of noise and collisions on latency for CSMA networks

Figure 7.28: Effect of $\lambda$ on noise with CSMA networks
7.4.5 Solving CSMA with collisions by adding a collision state

The CSMA type protocol including burst noise and collisions was analyzed in the previous section. The number of collisions was modeled by extending the service time according to the probabilities that hidden node and vulnerable time collisions would occur. The same approach is used in this section, but the collisions are modeled by adding another state probability. The state diagram shown in Fig. 7.16 is now expanded to that shown in Fig. 7.29. The arrival rate for this new state is equal to the arrival error rate, given by the sum of eq. 7.59 and 7.64. This rate can be reduced to that of eq. 7.86. The departure rate out of the collision state must be equal to that of the arrival rate, because the system is assumed to be in equilibrium. The state transition matrix can now easily be set up using a table, as shown in Table 7.1. All the different state probabilities are written on the left as well as at the top. Taking the rows one at a time, the transition rates can now be obtained directly from Fig. 7.29, by writing down only the processes that influence the state in that specific row. Only departure rates are taken into account, and they are written in the source state’s column. Note that departures that leave the state under inspection are positive, while arrivals are negative.

\[ \lambda_{\text{error}} = \lambda_{\text{error hid}} + \lambda_{\text{error tv}} = 2\rho(n\lambda + (N-n) \gamma) [\delta_{\text{hid}} + (1-\delta_{\text{hid}})t_e] \]

The effect of collisions on this model is shown in Fig. 7.30. Due to the complexity caused by the dynamic arrival rates, the utilization is not re-adjusted to compensate for retransmissions, as with the previous model. The effect of burst noise on this model is therefore less significant than with the previous model. This decrease in latency is only visible when high arrival rates are used. The utilization for this model is shown in Fig. 7.31. Note that both the noise and the no burst noise graphs are given by the same line and, therefore, no increase in utilization.
with an increase in $\lambda$ is visible, as it was with Fig. 7.18. The effect of hidden nodes is also much more dramatic using this model and can be ascribed to the non-linear approach used. The dynamic arrival process directly influences the utilization, which is not adjusted during simulation. The utilization is assumed to always be constant, independent of the number of stations in the system. The utilization is always determined using the maximum number of new arrivals. Even though this model does not take multiple retransmissions into account, the error arrival rate always enters the system when the utilization is at its worst and, therefore, the effect of collisions is much more severe. The effect of collisions and burst noise on latency, using this model, is shown in Fig. 7.32.

### 7.5 Simulink Model

The mathematical model used to analyze the particular CSMA protocol is based on numerous assumptions and is network specific. All stations are assumed to be distributed evenly throughout the available area. Changes in the network topology or communication strategy could require a completely new analysis. The protocol used in the mathematical analysis is also fixed and not very dynamic. The modeling is now approached from a more dynamic angle, that of a real-time simulink model. The idea is to design a simulink library block, which would react the same as an RSB. This block can then be duplicated and connected using different topologies to determine the latencies. No assumptions would be needed to determine the multi-hop effect, which could be modeled directly.
Chapter 7 — Performance Predictions

Figure 7.30: Effect of collisions on latency using the added state approach

Figure 7.31: Utilization using the added state approach
Figure 7.32: Effect of noise and collisions on the latency of a network using the added state approach

7.5.1 Basic design of the RSB block

The design process is simplified by using the SimEvents toolbox, which consists of numerous basic queueing blocks such as event sources, servers, queues, etc. The basic layout of the simulink library block is shown in Fig. 7.33. As can be seen, the model has two input and two output ports, to allow for duplex transmission if dual communication channels are available. If only one channel is available, then the output switch can be set to only use one of the channels. Also note that both the receive ports have their own payload buffers, as supplied by the hardware. The Flow control block tests whether received data comes from a valid source or not. If the received data is invalid, it is dropped.

Figure 7.33: The basic RSB simulink library block
The *Hardware* block simulates all the processing done by the microcontroller and can be expanded to that shown in Fig. 7.34. The *New A/D events* block models all new arrivals read from the analog to digital port and can be expanded to that shown in Fig. 7.35. New events are generated with exponentially distributed inter-arrival times, which are dependent on the arrival rate $\lambda$. The PDF of new events is therefore modeled by the Poisson distribution. The *Exponential Distribution* block was derived from a block found in the $M/D/1$ queue demo supplied by MATLAB (sedemo_md1). Every new event consists of 9 attributes which are set at event generation. The attributes used and their individual purposes are:

- **Payload** - Every packet contains a random payload value which is used to determine the direction of data flow. This field, in collaboration with the rest of the routing information, is also used to uniquely identify each packet. If a packet is being transmitted in the direction of the BS, then the payload is positive. If data is sent to a higher level station, the payload is negative. The value of the payload is used in the *Flow control* block to determine if the source is valid, by checking the data flow direction.

- **SourceID** - Every station has a unique ID, consisting of a level and a station. The first number represents the level, while the second number represents the station’s number on that level. If a station is the first station on level 3, then the ID would be 31. The SourceID is set to that of the station which generates the event. Note that the ID used can only support a maximum of 9 stations per level and also only 9 levels. This could very easily be expanded if required.

- **MSG_Type** - This attribute identifies the type of packet sent. Three message types exist and are shown below with their respective number.
  1. MSG - New messages sent.
  2. ACK1 - An acknowledge message sent back if a message is forwarded.
  3. ACK2 - An acknowledge message sent back by the destination station.

- **DestID** - The unique ID of the destination station is set in this field.

- **ID** - The last station to send this message is set in this field. When a new event is triggered, this value is the same as that of SourceID. If the message has to be forwarded, the station which forwards it, would put its own unique ID in this field and send an ACK1 message to the station whose ID was in this field on reception.

- **TimeStamp** - This field holds the time when the event was generated. This time is used to determine when a message has to be retransmitted due to expired ACK2 TTL timers. If a message is retransmitted, that new time is set in the TimeStamp attribute.

- **T2** - This attribute holds the time when the message was received at a station. This attribute is used to determine when messages have to be dropped or retransmitted due to an ACK1 message not being received. This attribute is set to 1 initially.
• **count** - This attribute is set to 0 whenever a new event is generated. When a station receives a message, this field is set to the number of messages already received by that station. This is used to prevent duplicates of the same message being added to a station’s ACK stack.

• **RetranCount** - Every message is allowed to be retransmitted only a limited number of times before it is dropped. This attribute counts the number of retransmissions.

![Figure 7.34: The Hardware block](image)

![Figure 7.35: The New A/D events block](image)

All messages (new messages, retransmitted messages and messages received from other stations) have to go to the *RX module* block, which is shown in Fig. 7.36. The *MSG control* block divides all the messages into two categories. The first category contains all messages destined for this level, while the other category only contains messages that have to be forwarded. Packets not destined for this station are now dropped from the first group, while the remaining packets are divided into MSG, ACK1 and ACK2 messages. All messages received by this station (MSG_Type = 1) now automatically generate an ACK2 message in the *PROCESS MESSAGE* block, which is sent back to the source. The messages that were not destined for this level are sent to the
Repeater module block which then forwards this message. This block also generates the ACK1 message that has to be sent back to the previous station when a message is forwarded.

![Figure 7.36: The RX module block](image)

Messages from the RX module block now move to the ACK management block. This block manages the ACK stack and is responsible for retransmission and dropping of packets due to TTL expirations. Again the messages are divided into MSGs and ACKs. This is done because the Level-2 M-file S-Function block has separate inputs for ACK and MSG packets. Note that the Enable Gate block prevents new data from being transmitted if the ACK stack is full. Every new packet is counted and that number sent to the Level-2 M-file S-Function block, to prevent duplication of data if new arrivals do not occur during the next sample period. The Level-2 M-file S-Function block was implemented using a S-Function. Before this type of function can be written, background knowledge on how simulink works is required. For more information on S-functions, please refer to [Mat07]. An explanation of the state transitions which occur during simulation is given in Fig. 7.38 (figure taken from [Mat07]).

![Figure 7.37: The ACK management block](image)

From Fig. 7.38 it is clear that the first step required is to initialize the block. Note that initialization only occurs once, at the beginning of the simulation, while the rest of the steps are executed during every time step. During this initialization, all the block’s interfaces and port
properties have to be set up. The block uses 6 dialog parameters to set the stack properties. All are set via the settings dialog shown in Fig. 7.45. The block also has 4 input ports and 5 output ports. Variables determined during every simulation loop in a S-function, are lost and can therefore not be used in the next loop. Every stack entry and the network statistics must be available at all times and have to be stored in memory. Normal variables can therefore not be used and a different strategy is required. To overcome this problem, Dwork variables are used. Work vectors are blocks of memory kept by each instance of a S-function, as requested by that S-function. These vectors are used, rather than global variables, to prevent data from being overwritten by another instance of the same block. Different types of work vectors exist, but the data type (D) is used. All these methods used to interface with the S-function block are listed below with their individual vector lengths in square brackets. All values are integers.

- The dialog parameter (DialogPrm) are:
  1. Stack depth for ACK1 [1]
  2. Stack depth for ACK2 [1]
  3. Number of attributes accompanying each packet [1]
  4. TTL for ACK1 messages [1]
  5. TTL for ACK2 messages [1]
  6. The station’s unique ID [1]

- The input ports are:
  1. Data TX [DialogPrm(3)]
Chapter 7 — Performance Predictions

2. Data RX [DialogPrm(3)]
3. Count TX [1]
4. Count RX [1]

- The output ports are:
  1. Message to be retransmitted [DialogPrm(3)+1]
  2. Relevant stack info [6]
  3. ACK2 data out [DialogPrm(3)+1]
  4. Full stack indicator [1]
  5. Messages to be forwarded [DialogPrm(3)+1]

- Dwork variables used:
  1. Dwork(1) - Contains all the station statistics [19]
  2. Dwork(2:DialogPrm(1)+1) - Stores all the ACK1 stack entries [DialogPrm(3)]
  3. Dwork((DialogPrm(1)+2):(DialogPrm(1)+DialogPrm(2)+1)) - Store all the ACK2 stack entries [DialogPrm(3)]
  4. Dwork(DialogPrm(1)+DialogPrm(2)+2) - Contains the data that has to be retransmitted [DialogPrm(3)]

- Layout of the Dwork(1) variable:
  1. Current ACK1 stack depth
  2. Previous TX number - Used to identify new packets
  3. Previous RX number - Used to identify new packets
  4. Number of packets lost
  5. Number of attempts to pop an empty stack
  6. Packets lost due to ACK1 not received
  7. Packets lost due to ACK2 not received
  8. Current ACK2 stack depth
  9. Number of ACK1 messages received
  10. Average wait time for ACK1
  11. Total ACK1 wait time
  12. Number of ACK2 messages received
  13. Average wait time for ACK2
  14. Total ACK2 wait time
  15. TX MSG number - Used to identify new messages to FWD
  16. RETRAN number - Used to identify packets to be retransmitted
17. Previous SourceID - Used to identify new packets
18. Previous TimeStamp - Used to identify new packets
19. ACK FWD number - Used to identify ACK messages to FWD

The S-function block is set to use a sample time inherited from its neighbouring feeding blocks. With the S-function used, no direct feed through loops exists and no derivatives or zero crossings have to be calculated. With the sampling time set to “inherited”, no updates of discrete states are needed either. The only state shown in Fig. 7.38 used, is that of “calculate outputs”. Data received on the Data RX port contains all messages to be sent to other stations. These messages are either messages sent from this station, in which case they should be added to the ACK1 and ACK2 stacks, or they are messages that are only forwarded, in which case they will only be added to the ACK1 stack. Before a message can be added to any stack, its validity has to be checked first, and it should also be checked that the message is not already in the stack. If one of these checks fails the message is dropped, otherwise it is added and sent out via the FWD MSG port. To indicate that a new message is available to be transmitted, the TX MSG number (Dwork(1).Data(15)) is increased. This number is always added to the MSG forward output port (OutputPort(5).Data(DialogPrm(3)+1)) and then used in the MSG FWD block to generate the message to send. Data received at the Data RX port contains all ACK messages received by this station and also the ACK2 messages that have to be returned to the original source station. On arrival of data at this port, validity checks are performed before a message is analyzed. If this station is the destination for a packet, the stack is scanned through to find the copy of this packet. If a copy is found, it is removed from the stack according to the type of message received. It should just be remembered that if an ACK2 message is received, the ACK1 message should also be removed, since the ACK2 indicates that the message has been received successfully. If an ACK2 message is received, but this station is not the destination, only the ACK1 stack would be scanned to see if the accompanying ACK1 message has been received, otherwise it would be removed here. This ACK2 message would then be sent to the ACK2 DATA OUT port and the ACK FWD number (Dwork(1).Data(19)) should be increased and added to the ACK2 FWD output (OutputPort(3).Data(DialogPrm(3)+1)). It should also be remembered that this block is also in control of TTL timers and retransmissions. After processing of new data on the RX and TX ports, the ACK stack is scanned to see whether any of the TTL timers of the station’s entries have expired. If such a packet is found, the entry is copied to the retransmit Dwork variable (Dwork(DialogPrm(1)+DialogPrm(2)+2)) and the RETRAN number (Dwork(1).Data(16)) is increased. This data is then send to the RETRAN output port together with the RETRAN number. The RETRANSMISSION block, shown in Fig. 7.39, uses this data to generate a new event with the same attributes. A flow chart of the implemented S-function is shown in Fig. 7.40, giving a more detailed analysis of the block.

All data that has to be transmitted, leaves the Hardware block and enters the Output Switch shown in Fig. 7.33. The channel used can be set with this block. During the final simulations of this thesis, both channels were used and chosen at random by this switch. The CSMA block
Figure 7.39: The Retransmission block

is shown in Fig. 7.41 and uses the same exponential distribution block as is used with the new event generation. This block has to sniff the communication channel to see whether data is already being transmitted or not. The communication channel is simulated by a single server with no queueing capacity, as shown in Fig. 7.42. The CSMA block has a queue which provides an output indicating whether the block has pending entities (pe) or not. A pe value of 1 indicates that messages are queued up and therefore the channel is busy. This is then used to indicate a busy channel and a back-off time is set using the exponential distribution.

7.5.2 Design of the communication channel

The communication channel is modeled using a single server and no queue. It must be remembered that all data transmitted on the same frequency in the same area, must use the same communication channel. In the case of multi-hop systems the communication channel has to take data from both transmission sides if they try to transmit to the same station. With different topologies, special care should be taken when modeling the communication channels. These channels are different for each topology. The communication channel for the network shown in Fig. 7.43, with 4 level 2 stations and one base station, is shown in Fig. 7.42. The switch and the pulse generator in Fig. 7.42 are used to model burst noise. The pulse generator’s period and pulse width can be varied to model different noise levels. Note that the arrival process has exponential inter-arrival times, while noise is generated using a periodic signal. The number of errors caused in the communication channel is, therefore, completely dependent on instantaneous arrivals. This means that the number of errors generated for exactly the same network can be varied, by simply changing the seed (see Section 7.5.4 for more information on seeds) of the new events generator block.
Figure 7.40: FlowChart of ACK-control S-function
7.5.3 Design of the BS

The base station is much easier to model than the router station boards. The base station does not generate its own events, retransmit data or have an ACK stack. The design of the base station is shown in Fig. 7.44. Data received is tested for validity in the BS RX module block and an ACK2 message is then sent if the data is valid. All arrivals are also counted and exported via the info port.

7.5.4 Network and simulation settings

All the different settings needed to model the network can be set using the settings dialog (see Fig. 7.45), which can be obtained by double clicking the settings block shown in Fig. 7.43, and the RSB dialog (Fig. 7.46), which can be obtained by double clicking the RSB block. Each station’s sector, level and level number are set via the RSB dialog block. The sector option is not used in this thesis, but allows for channel immunity if each sector uses a different frequency. A seed is used as a starting sequence when generating random numbers. The seed option is,
Figure 7.43: Simulink network with 4 stations and one BS no multihop.
Figure 7.44: The Base station block

therefore, used to allow each station to generate new events using different random sequences. If high collision rates are required, the seed for all stations could be set equal. The two check boxes are used to enable event generation and forwarding respectively. Stations can easily be disabled and enabled using these two options. The available settings for the settings dialog block are:

- Arrival rate - The arrival rate of new packets per station (\( \lambda_{\text{station}} \)).
- ACK1 stack depth - The maximum number of ACK1 messages to keep in the ACK1 stack.
- ACK2 stack depth - The maximum number of ACK2 messages to keep in the ACK2 stack.
- TTL for ACK1 - The maximum time to wait for an ACK1 message before dropping or retransmitting the packet, depending on the number of retransmissions already undergone. It should be noted that an ACK1 message must be dropped before a packet is retransmitted due to an ACK2 TTL expiration, otherwise multiple entries could exist.
- TTL for ACK2 - The maximum time to wait for an ACK2 message before dropping or retransmitting the packet, depending on the number of retransmissions already undergone.
- CSMA back-off time - The mean time to back-off if the channel is busy.
- TX queue depth - The number of messages that could be kept in the TX queue, while the communication channel is busy.
- Rise time - The time the transmitter needs to charge up, before transmission could begin.
- General processing time - This time includes data processing and all other forms of processing needed during packet reception and transmission.

7.5.5 Simulations and results

Numerous simulations were run to compare the simulink model with the mathematical model and the actual measured results. Throughout simulations the ACK1 and ACK2 stack depths
Figure 7.45: Settings dialog used to set the simulink simulation parameters

were set to 10 and the TX queue depth was set to 5. The rise time and the general processing time was also kept constant at 650\(\mu s\) and 500\(\mu s\) respectively. A typical example of the results obtained using this model is shown in Fig. 7.47. These results were obtained by simulating the network shown in Fig. 7.43, with \(\lambda_{\text{station}} = 1\), \(TTL_1 = t_{BO} = 120\text{ms}\). Burst noise was obtained by setting the pulse in the communication channel block to a period of 0.05s and a width of 4\%. The latency graph shows the individual latencies of the different stations, while the noise graph shows the communication channel noise. It can be seen from the graphs that the noise affects each station differently. Note the increase in latency when packets are lost, causing retransmissions. By looking at the graph showing the number of packets lost, it can clearly be seen that one lost packet has a much more severe effect on the network if the number of packets sent is small. As the time progresses, the total number of packets transmitted increases, and the effect of a single packet being lost decreases. From these two graphs, it is clearly visible that the network stabilizes and converges to a definite stable state. It should be noted that collisions were not incorporated into the communication channel block, but can be added if needed.

The results obtained from the simulink model were processed and are shown in Fig. 7.48 to 7.50. The first figure shows the effect that the number of stations has on the latency of the network. Note the additional increase in latency, with the increase in arrival rate, for the networks with more stations. The minimum time needed to transmit 1 packet successfully is approximately 14\(\text{ms}\). The maximum number of packets which could be serviced during 1s, if no losses or collisions occur, is approximately 71. The maximum arrival rate for the network with no errors is therefore approximately 71 packets per second. The increase in the number
of stations decreases this maximum arrival rate by the same fraction. The network is not error free, causing retransmissions to occur, which increase the network arrival rate even further. This increase in the number of stations explains the dramatic increase in latency shown in Fig. 7.48 for the 4 station scenario.

The previous figure does not take burst noise into account. The effect burst noise has on latency is shown in Fig. 7.49. The same response is obtained as with the mathematical model. Note how the noise is not constant, as with the mathematical model, but rather a running average dependent on time and the arrival PDF. The noise is modeled by dropping data packets according to the “on time” of a periodic pulse signal. If no data is being transmitted during the “on time”, no losses are generated. The noise is therefore directly dependent on the arrival seeds, as explained in Section 7.5.2. The last figure (Fig. 7.50), shows the effect that multi-hop has on latency. Note the step response in latency for the increase in the number of hops.

It must be stated that the simulink model cannot be used to model networks with very high arrival rates, which would cause the latency to strive to infinity. The reason for this is that the simulink model models a real time network and therefore has finite buffer space and limited processing capacity, as with the actual hardware. The mathematical model does not take buffer space or blocking into account. With the simulink model, retransmissions could easily cause more retransmissions. Even though the long term arrival rate is less than the service rate, the system could still lose all its packets if the instantaneous arrival rate during a certain period is
too high. Abnormally high instantaneous arrival rates (such as burst arrivals) could initiate a snowball effect which could undermine a complete network. The seed of the individual stations plays an enormous role and could cause two completely different results for the same arrival rate, by only changing the instantaneous arrivals. This random arrival process poses the same threats to this model as with the actual hardware and, therefore, multiple simulations with different seeds should be compared before the results can be used. Another problem with the simulink model, is that it cannot run for longer than 30 seconds of simulation time (note, not normal time), before it starts adding additional time to the latencies. This time is relatively long, compared to the time needed to send one message. This time constraint causes the latency obtained by this model to converge for networks with relatively high arrival rates, but tends to have quite a big ripple result for low arrival rates.

7.6 Comparison

The two protocols discussed in this chapter, are that of RRP and CSMA. The RRP was solved using two different approaches. The difference between the results obtained by the conventional method and by the queueing method is shown in Fig. 7.8 (see Section 7.3 for more detail). The second protocol, that of CSMA, was modeled using three different approaches. The first analysis is based on a state transition matrix where both the collisions and noise are modeled by
extending the service time. The second case also uses a state transition matrix, but an added state is introduced to model the collisions. The burst noise is still modeled by extending the service time. The third analysis consists of a simulink model.

The latency difference between the RRP protocol and the CSMA protocol is shown in Fig. 7.51. Note how the RRP protocol’s latency is constant, independent of service demand. For low data arrival rates, the CSMA latency is much better than the RRP latency, but deteriorates as the arrival rate increases. This latency increases even more with the addition of burst noise and collisions. Note how the intersection point between the CSMA graph and the RRP graph shifts to the left as the burst noise is increased. The prerequisite for the network designed was not high data traffic, but low cost. For low data traffic networks the CSMA protocol is a better option, especially if it is taken into account that the RRP latency would increase even more with an increase in the number of stations.

A comparison between the expected latency for the CSMA model derived in Section 7.4.4, and the model derived in Section 7.4.5, is shown in Fig. 7.52. The two models follow each other very closely when the arrival rate is low. The maximum arrival rate which could be supported by an error free network was derived earlier as approximately 71 packets per second. According to the first CSMA model, the maximum arrival rate which could be supported when 10% noise and collisions are added is 3.2 packets per second per station (note there is 10 stations). This represents about 45% of the maximum ideal rate. For the figure shown, the latency values for the two models are within 10% of each other while the arrival rate is less than 1. These two
Figure 7.50: Effect multi-hop has on the simulink model where $\lambda_{\text{station}} = 0.5$, $t_{BO} = 20\text{ms}$ and $TTL_1 = TTL_2 = 60\text{ms}$

models, therefore, have a close relationship while the arrival rate is low, but tends to deviate as the arrival rate increases. The added state model has a tendency to be affected more drastically in the presence of noise than the other CSMA model.

As discussed earlier, the stability of the simulink model is completely dependent on the arrival distribution, as generated by the seed of each station. Instantaneous arrivals could severely increase or decrease the system latency. A comparison between the predicted latencies of the RRP, CSMA and simulink models is shown in Fig. 7.53. Note how the CSMA latency predictions for both the mathematical and simulink models are superior to that of the RRP model at low arrival rates. The simulink graph is not a straight line, compared to the mathematical model, as the results are completely dependent of instantaneous arrivals. The dip in latency for the simulink model, at arrival rate of 10 packets per second, could have been caused by a very low instantaneous arrival rate during the time of measurement. Better values could be obtained by changing the individual seeds of the stations.

A comparison between the CSMA model and the simulink model for a network with a multi-hop layout is shown in Fig. 7.54. The latency predicted by the simulink model varies around the 2 hop average. This is caused by a higher number of level 3 stations transmitting data than that of the higher level stations, therefore moving the average down, because the mean number of hops decreases. This average could be increased by increasing the seed of the higher level stations, or by decreasing the seed of the lower level station. The obvious solution would be to make the seeds
of all the stations equal, but this would cause collisions during each transmission, as they would then all transmit during the same time. The transmission buffers of the lower level stations forwarding the data would also tend to overflow due to the large amount of data that has to be forwarded. This simulink model used during this simulation used dual communication channels, because that is the system implemented by the hardware. The CSMA mathematical model can be adjusted to use dual channels, by changing the multiplication factor of the minimum time needed to send one message, from 2 to 1.25. The multiplication factor of 2 compensates for the ACK message needed when sending data. This means that twice the minimum time needed to transmit one message is required by the communication channel before another station is able to start using the channel. If dual channels are used the amount of data being sent is the same, but the traffic density decreases and therefore the channel busy times decrease. To motivate this, an example of a packet being transmitted is used. If a packet is transmitted via channel 1, that channel would be busy with probability of 1. Any other station could transmit data during the same time via channel 2. On reception of the packet an ACK message has to be sent back. This message could be transmitted via either channel 1 or 2 with equal probability. Other stations that might also try transmitting data could also be using either of the communication channels. The probability of another station and this station transmitting data at the same time, using the same channel, is therefore 0.25 (0.5 × 0.5). If both stations try to transmit via the same channel, only one could succeed, while the other would have to wait. The total time the communication channel is busy with a packet is therefore $1.25h_{mean} \left[ t_{\text{min}} + δt_{\text{TTL}} \right]$ seconds.

Figure 7.51: Latency comparison between CSMA and conventional RRP
Both RRP and CSMA were analyzed mathematically in this chapter. The main focus was on CSMA. RRP were analyzed both analytically and by using queueing theory. Both RRP methods provided similar results, especially when the number of stations is high.

An existing CSMA model, based on queueing theory and a state transition matrix, were extended to include the effect of collisions. A statistical study of collisions (hidden nodes and rise times) was performed for this purpose. The collisions were then added to the current model by either adding an additional state or by extending the service time. The results from both these models are quite similar, especially under low load, which is what we are interested in.

It was found that the RRP’s performance is superior under heavy load, but CSMA is better under low traffic load, as expected (see Fig. 7.51). The use of a simulink model to predict performance was also introduced in this chapter. The results obtained from this model are relatively accurate, but the extent of this model is limited by simulation constraints.

**Figure 7.52:** Comparison between the CSMA model and the added state CSMA model, for a network with 10 stations and $t_{BO} = 20 ms$ and $t_{TTL} = 60 ms$
Latency difference between the CSMA, conventional RRP and simulink model for a network using 4 level 2 stations and no additive burst or collision noise

Figure 7.53: Comparison between the latency of the CSMA, conventional RRP and simulink model for a network with 4 level 2 stations and no additive noise ($t_{BO} = 20ms$ and $t_{TTL} = 60ms$)

Mean latency vs $\lambda_{\text{station}}$ for a network with 10 stations and no burst noise

Figure 7.54: Comparison between the latency for the CSMA and simulink model, for a network with 1 level 2, 4 level 3, 4 level 4 and 1 level 5 stations ($N_{\text{mean}} = 2.5$, $t_{BO} = 20ms$ and $t_{TTL} = 60ms$)
Chapter 8

Measurements and Results

The main purpose of this chapter is to compare the results predicted by the various CSMA models derived to the actual measured results. The measured RF results are also shown and discussed in this chapter. Other relevant measured values are shown in Table 8.1, while some important values required for the simulations are shown in Table 8.2.

8.1 RF measurements

The nRF905 transceiver has numerous RF related settings. The two settings of utmost importance are that of the centre frequency and RF power. The hardware implemented uses 2 communication channels and, therefore, two isolated frequency bands must exist. The transceiver chip used supports up to 7 available channels. The inter-channel isolation between 4 channels is shown in Fig. 8.1. Note how channel 3 is left out to show that the noise remains constant at -40 dBm. Also note how the channels do not overlap. The two channels used by the software are those of CH1 and CH2. Plans were initially made to also use CH3 and CH4, to reduce the effect of collisions. The idea was that stations would use different frequencies depending on the levels they were on. The option of using different channels for different sectors was also examined. Due to the complexity of new station registrations, these ideas were not implemented.

With the nRF905 transceiver 4 different power settings exist. They are 10 dBm, 6 dBm, -2 dBm and -10 dBm respectively. The transceiver used can be bought as a complete packet, with matching network and oscillator built in, and uses a standard dip-24 footprint. With the

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{dd}$</td>
<td>Voltage supplied by the LM117 regulator</td>
<td>3.27V</td>
</tr>
<tr>
<td>$I_{in}$</td>
<td>Input current drawn by the RSB</td>
<td>6mA</td>
</tr>
</tbody>
</table>

Table 8.1: Measured values
transceiver module used a guaranteed output power of at least 6 dBm on the 10 dBm setting can be expected. The different power settings are shown in Fig. 8.2. Note the relative weakening in power for the different settings. Also note that the minimum power of 6 dBm is achieved for the 10 dBm setting.

### 8.2 Latency measurements

The derived communication strategy was implemented on the hardware. The hardware is accessed via a GUI which communicates through the PC’s serial interface. To analyse the network, the GUI was expanded to incorporate a virtual analog to digital interface. Events are generated on a PC with inter arrival times given by eq. 8.1. The preferred mean time is set in the program ($t_{\text{mean}}$) and is divided by two with a pseudo-random value between 0 and $t_{\text{mean}}$ added subsequently. This gives a truly random inter arrival time ranging from $t_{\text{mean}}/2$ to $3t_{\text{mean}}/2$. Each PC used, uses a different seed to generate the pseudo-random values to assure unique generation. These events are then sent to the RSB, which is then detected as if it is a packet generated by its own ADC port. By adding this virtual interface, no changes to the onboard PIC program are required for testing purposes and the RSB can use its normal communication strategy. All stations used for testing should therefore be connected to a PC. The payload of the events generated contains a packet number indicating the number of packets already sent from that PC. After transmission of an emulated packet, the packet is sent back to the GUI, where it is logged in a database containing all the transmission information added by the RSB. A time stamp is also added. When the RSB receives a valid ACK for an emulated packet, it sends the data to the PC. The matching entry stored in the database at transmission, is then filtered out using a SQL query. The difference between the time of reception and time of transmission is determined, as well as the data rate and both are added to the database. It should be noted that the timestamps for the transmit and receive packets are added by the PC on reception of the data from the RSB. This is done before filtering and database handling to prevent additional

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR</td>
<td>50kbps</td>
<td>Data rate (Manchester encoding)</td>
</tr>
<tr>
<td>$t_B$</td>
<td>20us</td>
<td>$\frac{1}{BR} = \frac{1}{50000}$</td>
</tr>
<tr>
<td>$t_{\text{preamble}}$</td>
<td>200us</td>
<td>10 bits @ 20us per bit</td>
</tr>
<tr>
<td>$t_{\text{postamble}}$</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$t_{\text{prop}}(\tau)$</td>
<td>3.3us</td>
<td>$\frac{\text{distance}}{c} = \frac{1000}{3}$</td>
</tr>
<tr>
<td>$t_{\text{gen}}$</td>
<td>500us</td>
<td></td>
</tr>
<tr>
<td>$b_{\text{preq}}$</td>
<td>136 (17 bytes)</td>
<td>15 payload bytes, 2 CRC bytes and 4 address bytes</td>
</tr>
<tr>
<td>$b_{\text{prep}}$</td>
<td>168 (21 bytes)</td>
<td></td>
</tr>
<tr>
<td>$t_{\text{rise}}$</td>
<td>650us</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.2: Important simulation variables
query and processing times from being added to the packet latency. The GUI also determines the network efficiency using eq. 8.3. This equation states that the best possible efficiency for this network is 10.56%. This deteriorates as retransmissions and packet losses occur. Multi-hops do not influence the ratio, because both the number of successful packets and total number of packets are scaled by the same hop factor.

\[ t_{\text{inter}} = \frac{t_{\text{mean}}}{2} + \text{rand}(t_{\text{mean}}) \]  

\[ \eta_{\text{best}} = \frac{\text{Packets}_{\text{successful sent}} \times T_b \times \text{Bytes}_{\text{data}}}{\text{Packets}_{\text{all}} \times T_{\text{total transmission}}} \]

\[ = \frac{(\text{Packets}_{\text{successful sent}})(20\,\text{us})(17)}{\text{Packets}_{\text{all}}[20\,\text{us}] + (2 \times 200\,\text{us}) + (2 \times 650\,\text{us}) + 2\tau} \]

\[ = 0.1056 \times \frac{\text{Packets}_{\text{successful sent}}}{\text{Packets}_{\text{all}}} \]

The actual measurements were very difficult to obtain because all stations needed a PC to be able to generate new data. The stations were set up in a large laboratory which had enough PCs. The background RF noise in this laboratory was very high due to numerous other electronic devices such as power supplies, oscilloscopes, signal generators and PCs. People were also continuously moving between the antennas creating obstacles and multipaths. The physical size of the laboratory also made it very difficult to test the multi-hop effect, seeing that all stations could cover the whole area. To solve this problem, the transmission radius of each RSB was
Typical measured results are shown in Fig. 8.3 to 8.6. The first figure shows the instantaneous and average latency measured by one station in a network where 4 level 2 stations are transmitting data. An arrival rate of 1 packet per second per station was used ($\lambda_{\text{station}} = 1$). To determine the total average latency of the network, the average of all four stations’ averages has to be taken. Note the spikes in latency for packets that were retransmitted ($t_{TTL} = 120 ms$). Also note how only a couple of retransmitted packets can influence the average latency. The accompanying data rate measured at the same station is shown in Fig. 8.4. Note that this graph does not represent the baud rate, but only the actual data sent. To determine the actual throughput, the value obtained in this figure should be scaled by $\frac{38}{17}$ to include the headers, routing information and CRC padding. The value should also be doubled, to add the ACK packet sent back. The total throughput can therefore be obtained by multiplying the values in Fig. 8.4 by 4.47. If the highest value of 9.067 is used, a throughput rate of 40.53 kbps is obtained. This is quite a good value if the fact is kept in mind that the maximum rate supplied at this frequency by the nRF905 chip is 50 kbps. This rate does not include the time required to control the transceiver (processing and communication between the PIC and the transceiver via SPI) or the additional rise times.
The latency measured for a network with 1 level 4 station is shown in Fig. 8.5. This example is supplied to show the difference between the no-hop case, shown earlier, and the multi-hop case. It should be noted that the shortest time possibly required to transmit a 3 hop packet, is approximately 42 ms (3 × 14 ms). The latencies shown in Fig. 8.5 that are less then 40 ms, represent the packets that missed a skip by sending the data directly between 2 stations that are further than one hop apart. The communication strategy used allows this, because no routing station information is added, but an XOR algorithm is used to determine the route. If this should happen, the skipped station would still receive the packet, but would not know that it had been skipped. It would therefore still forward the packet to the next station, which would already have received it. On reception at the next station, the station would check its ACK stack and find that it had already received this packet and would, therefore, drop the packet. The packets that skipped a hop, therefore, do not influence the network negatively at all, but merely reduce the latency for that packet. The routing tables are not affected by this at all. It should be remembered that this only happens with very poor links, otherwise those two stations would have been neighbouring stations. The data rate graph for this 3 hop case is shown in Fig. 8.6. Note the decrease in data rate when multi-hop is introduced. It should be stated that the link quality for this test was very bad and therefore the measured latencies are very high.

The previous measurements were taken prior to the optimum setting being tested for, and therefore the optimal back-off, TTL and service times were not used. These measured results were used only to show the basic effect of multi-hop, but mostly to show the format of the results.
Figure 8.4: Instantaneous and average data rate measured by 1 station in a network with 4 level 2 stations transmitting with $\lambda_{\text{station}} = 1$

measured by the GUI. The optimal settings will now be discussed.

- **Service routine**

Every packet that is transmitted is added to the ACK stack together with a time stamp used to determine when packets should be retransmitted. Packets that are waiting in the queue to be transmitted are also sent to the ACK stack, but their number of retransmissions is set to 0, indicating that they have not yet been transmitted. All these TTL timers have to be monitored simultaneously. An interrupt cannot be set for each entry because the PIC does not support that many interrupts. The only way to monitor the whole stack is to make use of a service routine. The optimal time between consecutive routines should now be determined. The shortest time possible should be used to prevent unnecessarily long additions to TTL timers. If this time is too short, it would halt the PIC operations such as ADC monitoring and serial communication. It must be kept in mind that this time should be shorter than that of the TTL time, to try and reduce the TTL error. The best service interval is that which produces the lowest latency and percentage packet loss, while still executing all its other tasks, such as ADC monitoring and data transmission via both RF and serial. The measured results obtained from numerous tests are shown in Fig. 8.7. Each value shown in this figure represents an average taken from 5 different tests run with the same settings. Note how the average latency increases with an increase in service interval time, while the percentage of errors decreases. The optimal time is taken as 40
Figure 8.5: Instantaneous and average latency measured in a network with 1 level 4 station

\[ \text{with } \lambda_{\text{station}} = 1 \]

ms. It should be stated that times less than 20 ms do not allow the station to perform all of its other tasks, while values larger than 60 ms are not permitted because the TTL used, as will be discussed later, is 60 ms.

- **Back-off**

The minimum time required to send one successful packet is 14ms. A reasonable mean for the back-off time should therefore be chosen between 5 ms and 30 ms. The back-off is implemented using one of the PIC’s interrupt timers. The low value of 5ms tends to cause problems when high arrival rates exist, because the microcontroller can no longer service all its procedures fast enough. It is recommended that a higher value be used for back-off. The effect the back-off mean has on the latency and percentage errors is shown in Fig. 8.8. Note, again, that the values represent an average taken over 5 independent tests. It is clear that both the latency and error percentage increases with an increase in back-off time. The minimum value that the PIC can handle without losing any functionality should be used. This value is chosen as 20 ms.

- **Time-To-Live (TTL)**

Looking at the measured latency in Fig. 8.3, it is clearly visible that the average latency for a single hop system would be between 20 ms and 30 ms depending on the noise level and packet arrival rate. This project is aimed at low congestion networks and therefore the arrival rate is assumed to be very low. Even when more stations are added, the arrival
rate is assumed to be low enough not to influence the latency. The chosen TTL should therefore account only for the link quality. To be safe the TTL is chosen as 60 ms.

The effect that an increase in arrival rate has on both the latency and the percentage of lost packets is shown in Fig. 8.9. The network used consists of 4 level 2 stations with the ACK stack service interval set to 40 ms, the mean back-off time set to 20 ms and the TTL set to 100 ms. It should be stated that each point on the graph represents the average for all 4 stations, where the average value for each station is obtained from 5 independent tests. Note how the average latency increases exponentially with an increase in $\lambda_{\text{station}}$. Also remember that the arrival rate shown on the bottom axis should be multiplied by 4 if the network arrival rate is desired. Looking at the number of packets lost during these tests, it is clearly visible that the link is very noisy. During these tests, 4 retransmissions were allowed before a packet was dropped as lost. The latency shown is not affected by lost packets because they are not taken into account when the average latency is determined. The latency for a lost packet is infinite and the average would therefore always be infinite if they were taken into account. Retransmissions that did not end in packet loss are taken into account, though. As the arrival rate increases, the number of collisions increase, causing the number of retransmissions to increase, which then causes the latency and the number of packets lost to also increase.

The effect of $\lambda_{\text{station}}$ on multi-hop networks is shown in Fig. 8.10. This network consists of only 1 level 4 station with $t_{\text{service\ routine}} = 40\ ms$, $t_{BO} = 20\ ms$ and $t_{TTL} = 40\ ms$. Note that the
Figure 8.7: Optimal service interval

TTL setting of 40 ms implies the no-hop situation, and should be multiplied by the number of hops taken, to determine the TTL setting for each level. The TTL time for a level 4 station is therefore 120 ms. If the number of packets lost for the multi-hop case is compared to that of the no-hop case, it will be seen that it is much higher. The number of packets lost in the multi-hop case generates a straight line that increases very slowly. This happens because only one station is used and collisions occur only between packets that are sent shortly after each other. With the arrival rates used, this should not happen. The only source for retransmissions is therefore burst noise, which remain fairly constant because the link quality remains the same. The increase in latency can be ascribed to an increase in the number of retransmissions.

8.3 Comparison between the simulated and measured results

In the previous section the setup procedure for the tests was explained, together with the basic effects of arrival rate on the network. In this section the measured results will be compared to the simulated results. A comparison for a network with 4 level 2 stations is shown in Fig. 8.11. The measured results and those of the simulink model are within 10% of each other over the whole range. For the simulink model noise was added to force retransmissions close to those measured. A comparison between the percentage of errors forced by the simulink model and the percentage of packets lost during measurements is shown in Fig. 8.12. Even though no packet loss occurred with the simulink model, the percentage of retransmissions is within the same range as that of the measured results. As stated earlier, 4 retransmissions are allowed before a packet is dropped. The number of retransmissions for the simulink model is roughly 5 times higher than the measured packet loss. The number of retransmissions in the simulink model could therefore easily have been the same as obtained with the measured results. The latency simulated seem
Figure 8.8: Optimal mean back-off time

to predict the measured values quite nicely. The number of predicted retransmissions is also completely within the expected range. The CSMA model also seems to predict the latency with good accuracy, if the amount of burst noise is decreased to 1%, while the added state CSMA model tends to deviate as the arrival rate increases (refer to the last paragraph of this section for an explanation for this).

After the no-hop measurements, the 1-hop case with 3 level 3 stations was set up and measured. The results obtained are shown in Fig. 8.13. The measured latency is clearly too high for the accompanying measured packet loss. After closely inspecting the network and the hardware, it was found that a defective power supply somehow damaged 1 transceiver on one of the RSB stations during the tests. Packets assigned to this damaged channel were logged as transmitted, even though the transceiver actually transmitted nothing. Channels are assigned at random and at least one of the 4 retransmitted attempts was assigned to the other of the two channels, which then managed to successfully transmit the message. Even though only a few packets were lost, the number of retransmissions was quite high, causing the very high latencies. Even with the other channel still working perfectly, this station could not be used because it no longer had the same duplex capabilities as the other stations. Only 4 stations were therefore available to do further tests with. This forced the use of only 2 level 3 stations. The comparison between the results for a network with 2 level 3 stations with $t_{BO} = 20\text{ms}$, $t_{service\ routine} = 40\text{ms}$ and $t_{TTL} = 60\text{ms}$ is shown in Fig. 8.14. All predictions are within 10% of the measured results during the period measured, but the simulink model and the added state CSMA model tends to deviate more and more as the arrival rate increases. The normal CSMA model clearly provides the best prediction. During this test the number of retransmissions was also measured. A comparison between the measured retransmissions and the retransmissions simulated with simulink is shown in Fig. 8.15. With the unpredictability of noise taken into account, the difference between the number of retransmissions is not that much. Note that no packet loss occurred at the simulink
Figure 8.9: The measured effect of $\lambda_{\text{station}}$ on the latency and percentage errors for a network with 4 level 2 stations (no multi-hop) with $t_{BO} = 20\,ms$, $t_{TTL} = 100\,ms$ and $t_{\text{service routine}} = 40\,ms$.

Figure 8.10: The measured effect of $\lambda_{\text{station}}$ on the latency and percentage errors for a network with 1 level 4 stations with $t_{BO} = 20\,ms$, $t_{TTL} = 40\,ms$ and $t_{\text{service routine}} = 40\,ms$. 
Mean latency vs $\lambda_{\text{station}}$ for a network with 4 level 2 stations and $t_{BO} = 20 \text{ ms}$ and $t_{TTL} = 110 \text{ ms}$

Figure 8.11: Comparison between the simulated and measured results for a network with 4 level 2 stations with $t_{BO} = 20 \text{ ms}$, $t_{\text{service routine}} = 40 \text{ ms}$ and $t_{TTL} = 110 \text{ ms}$

Comparison between the percentage of noise generated by the simulink model and that produced by the measured results

Figure 8.12: Comparison between the percentage of packets forced to be dropped by the simulink model and the percentage measured as lost
model, which explains the higher latency compared to the measured results. Lost packets are not taken into account when determining latency, because one lost packet would cause the latency to become infinite. If a packet is retransmitted 3 times, it would severely deteriorate the latency, while a packet that is dropped after 4 retransmissions would not affect the latency at all.

The measured latency for a level 4 station is compared to the predicted values obtained by the different models and this is shown in Fig. 8.16. The simulink model with added noise tends to follow the measured values closely. The percentage retransmissions and losses for both the simulink model and the measured results are shown in Fig. 8.17. With the 3 hop network under study the simulink model generates approximately 7% packet loss, while that of the measured results is approximately 21%. The percentage of retransmissions forced by the simulink model is roughly 77% which could at maximum have caused 19% packet loss. The simulink model’s latency should therefore be slightly superior to that of the measured results, which is clearly the case shown in Fig. 8.16. The amount of noise generated with the simulink model could be increased to obtain better results. The added state CSMA model with 40% burst noise and collisions also follows the measured results closely when the arrival rate is low. Only one level 4 station is used and therefore the effect of collisions is very small compared to the effect of the burst noise. With the added state model the utilization is not adjusted when retransmissions occur. This increase in utilization would increase the number of collisions and could also cause more retransmissions which would increase the utilization even more. This lack of increase in utilization causes the added state model’s latency to increase slightly slower due to noise than it does with the measured results. This explains why the added state model’s latency does not

Figure 8.13: Measured latency tests for 3 level 3 stations with 1 RSB only having one channel
Comparison between the measured results and simulated values for a network with 2 level 3 stations with $t_{BO} = 20\text{ms}$, $t_{service\ routine} = 40\text{ms}$ and $t_{TTL} = 60\text{ms}$

**Figure 8.14:** Comparison between the mathematical model, simulink model and theoretical measurements for a network with 2 level 3 stations transmitting ($t_{BO} = 20\text{ms}$, $t_{TTL} = 60\text{ms}$ and the service routine is set to $40\text{ms}$)

Comparison between the percentage retransmissions and packets lost for the simulink model and the measured results

**Figure 8.15:** Comparison between the percentage of packets lost and retransmitted for the simulink model and the measured results for a network with 2 level 3 stations
increase as much with an increase in arrival rate as does that of the measured results. The values obtained with the simulink model with no noise are also shown in this figure to show the effect of noise on the simulink model. In a no noise environment the expected measured results would have been in this region.

8.4 Conclusion

The protocol used is very difficult to model exactly, due to the numerous inter-connected variables that influence the performance of the network. These variables includes back-off times, arrival rates, noise levels, collisions, TTL times, network topology, etc. The ways in which these variables influence the network are not always the same and are also inter-dependent. For example, longer TTL times would cause fewer collisions in low arrival rate networks, while they would cause more losses in high arrival networks because of buffer overflows. Another big problem is that of burst arrivals, which could increase the arrival rate over a short time period to multiples of the actual arrival rate. These types of arrival can easily cause the simulink model or the actual hardware to fail. The simulink model is also limited to roughly 30 seconds of simulation time before data overflows start to cause problems.

Overall the simulink model seems to model the measured results best. The effect of noise is dif-
Figure 8.17: Comparison between the percentage of packets lost and retransmitted for the simulink model and the measured results for a network with 1 level 4 station

dicult to simulate with this model and therefore requires numerous simulations to obtain exactly the correct noise levels desired. The simulation itself also takes a very long time, especially in the presence of a high station count. The advantage with this model is that the same topology as that measured can be built and the link quality of each link can be set individually.

The normal CSMA model also makes accurate predictions, with the exception of multi-hop networks with a low station count. In this case the number of collisions is very low for low arrival rates and, therefore, predicts latencies that are too low. The effect is alleviated when the arrival rate increases, which increases the number of retransmissions due to noise. The effect of noise on the CSMA model is very pronounced because the utilization is adjusted to compensate for the increase in channel density due to retransmissions. This aggravation seems to be a good assumption.

The added state CSMA model predicts better latencies than the normal CSMA model does for multi-hop networks. The effect of noise is less aggressive with this model than with the normal CSMA model, because the utilization is not adjusted. The noise level used to model a network is closer to that measured when the same latency results are obtained. The effect of collisions is much more severe with this model than with the normal CSMA model. The latency predicted by this model for networks with very high arrival rates is, therefore, much too high due to collisions.

The added state CSMA model can best be used for multi-hop networks with low arrival rates
and low station counts. On the other hand, the normal CSMA model can be used for high arrival rate networks with high station counts. The network studied in this thesis features a low arrival rate and a large number of stations. This falls in-between the two categories as listed above and while the theoretical approach provides useful guidance, the most accurate results are, in general, obtained from the simulink emulation.
Chapter 9

Summary and Conclusion

9.1 Summary

In this thesis the possibility of realising a cost effective alternative for short distance data acquisition networking was studied. The research was based on an actual problem presented by an agricultural community in South Africa. A study of the actual terrain, using DEMs in collaboration with radio propagation software, revealed that the use of low power (10 dBm) transceivers would provide sufficient radio coverage of the given area, if an ad hoc network configuration were used. These low power transceivers reduce the cost of the monitoring stations significantly. The cost is further reduced by removing unnecessary interfaces and features, which are usually available on standard telemetry hardware.

In addition to sufficient coverage provided by the proposed hardware, controlling software is also required. Data communication must be implemented in terms of some suitable protocol. A CSMA based protocol was used for data communication. This protocol was thoroughly analyzed in Chapter 7 to derive a realistic and valid mathematical model of the protocol. The standard approach used to model CSMA protocols was expanded to incorporate a statistical analysis of collisions. The collision statistics were used to model errors caused by collisions. This addition provides a much more accurate statistically based prediction model, rather than the chosen-percentage method used in previous models. The previous model also did not adjust the network utilization when errors occurred, but rather determined the number of errors based on the utilization. The model derived in this thesis uses a recursive procedure to adjust the utilization according to the errors, while still incorporating error calculation based on the utilization. An increase in utilization would, therefore, increase the number of errors, which would further increase the utilization. This snowball effect is a typical phenomenon encountered in real life systems, which renders the approach of the model derived in this thesis much more authentic. The developed model was used to predict system performance and to determine optimal network settings.
A simulink model of this CSMA based protocol was also designed to verify that the model is, in fact correct. Both the simulink- and the mathematical models were compared with actual measured results to determine and verify the accuracy of the models.

9.2 Conclusion

This thesis proves that an ad-hoc network provides a definite solution for data acquisition networks. Not only does it reduce cost, by limiting the hardware requirements, but it also simplifies the network planning process and extends radio coverage by introducing multi-hop functioning. Radio propagation prediction software (Radio Mobile) was used to prove that an ad-hoc network would provide sufficient coverage.

A study of different communication protocols confirmed that a CSMA based contention protocol would provide the most suitable solution for this type of problem (one where the arrival rate of new events is not that high). The study was based on mathematical prediction models and theoretical knowledge of the respective protocols.

A current popular CSMA prediction model, based on queuing theory and a state transition matrix, were extended to include collisions. This provides a more realistic prediction model. Two solutions were presented. The first solution extends the service time to simulate possible added latency due to packet loss. The second solution introduces a new collision state to the current transition matrix. The results obtained from both these models were in line with actual measured results.

A simulink model of the network was also created as an alternative prediction method. Even though the model had some problems related to simulation time, the accuracy of the predicted results was acceptable.

The proposed hardware was implemented and the corresponding protocols added in the form of embedded software. Measured results corresponded to those predicted by the models.

9.3 Recommendations

Even though the hardware and software designed during this study provided an adequate performance, numerous improvements can be made. In this section some of the proposed adjustments will be briefly discussed, together with proposals for future work.

- RF interface hardware - The layout of the RF matching network described in this thesis varied somewhat from the proposed layout. This resulted in very high mismatch and reflection losses. The RF-chip is also very sensitive and requires proper grounding. It is
proposed that the complete RF modules manufactured by Polygon technologies be used to ensure performance. This would also simplify the replacement of RF modules if required, seeing that these modules are manufactured as a DIP.

- **Reduce header length** - The current header is 15 bytes long, while the data section consists of only 17 bytes. Each station in the network can be assigned a unique registration number together with its name. This number field can be used when referencing stations, rather than their actual names. The solution presented in this thesis is intended for small, short range networks and, therefore, the number of stations in the network would not be very high. If 1 byte were used for this ID number, the network would be able to support up to 256 stations. The 8 bytes used for the SID and DID fields can then be shortened to 2 bytes. The SLVL and DLVL fields represent the level of each of the stations. A level $n$ station requires $(n - 1)$ hops to reach its destination. It can be assumed that no more than 16 hops would be required to reach any station in the network. The 2 bytes required for these two fields can thus be reduced to 1 byte (4 bits each). By applying these changes, the header is shortened to 8 bytes. These adjustments were not applied to the protocol in this thesis, as the data requirements were not that high. The packet layout used in this thesis is easier to interpret by a user if a packet has been received.

- **Use FWD message as ACK** - Each message sent requires an ACK message to verify that the packet has been received successfully by a neighbouring station. If a packet has to be forwarded by a station it first sends an ACK message back to the previous station, before forwarding it to the next station. This redundant message can be removed by adapting the protocol to detect the forward message and use that as an ACK. Look at an example where station A sends a packet to station C via station B. With this new adapted protocol station A would send the packet to station B. When station B receives this packet, it would forward it to station C. Station A is within range of station B and would be able to receive the packet. From the packet header, station A would be able to determine that this was the same packet it had sent earlier, and would use the FWD packet as the ACK. Station C does not have to forward the packet and would reply with a normal ACK message. With this improvement the total traffic density can be greatly reduced. The removal of this now redundant packet and the resulting lower traffic density would decrease the overall system latency.

- **Include piggybacking** - Piggybacking is a term that refers to the process where data from another station is appended to a current message if sufficient room is available in the data payload. The fixed payload of this hardware makes it very difficult to include such a feature, because the messages are very long when compared with the total available data section. The network protocol could be adapted to allow piggybacking of short control messages, especially with ACK messages.

- **Improve the server and GUI** - There is endless room for improvement of the server. The server designed in this study is able to perform only basic tasks. Downlink features were
added, but minimal uplink features are available. The GUI should be able to control the output ports of the stations, to allow it to perform tasks such as opening and closing of valves and switching switches on and off. The server can also be extended to allow users to set up rules where some functions are performed in reaction to certain specified input. An example of this would be that the server would stop all pumps if a pipe burst. Another improvement to the server would be to use an online database, such as a mySQL server, rather than a stationary database.

- Optimizing the simulink model - The simulink model takes a very long time to simulate network performance and is limited to approximately 32 seconds of simulation time. Seeing that this model provides rather accurate predictions, it would be wise to try and improve it. This model contains various redundant blocks which can be removed, and other sections can be replaced by a simple s-function. A reduction in the number of blocks used by the model would reduce the simulation time. Closer examination of the model might also shed more light on the problem of the time limit. If this model could be adjusted to simulate over longer periods, it could be used to further study the effect of various topologies and network settings on the network latency. The block used to simulate channel noise should also be improved.
Appendix A

Hardware design Information

This appendix contains some relevant information needed to design the hardware. The complete documentation can be found on the attached CD-ROM. See Appendix B for more detail.

A.1 Design information on the nRF905 transceiver

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>PARAMETER (condition)</th>
<th>MIN</th>
<th>MAX</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{DD}$</td>
<td>Supply voltage</td>
<td>1.9</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>HIGH level input voltage</td>
<td>0.7$V_{DD}$</td>
<td>$V_{DD}$</td>
<td>V</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>LOW level input voltage</td>
<td>VSS</td>
<td>0.3$V_{DD}$</td>
<td>V</td>
</tr>
<tr>
<td>$I_{PD}$</td>
<td>Supply current in power down mode</td>
<td>2.5</td>
<td></td>
<td>uA</td>
</tr>
<tr>
<td>$I_{SPI}$</td>
<td>Supply current in SPI programming</td>
<td>20</td>
<td></td>
<td>uA</td>
</tr>
<tr>
<td>$f_{XTAL}$</td>
<td>Crystal frequency</td>
<td>4</td>
<td>20</td>
<td>MHz</td>
</tr>
<tr>
<td>BR</td>
<td>Data rate</td>
<td>50</td>
<td></td>
<td>kbps</td>
</tr>
<tr>
<td>$f_{CH433}$</td>
<td>Channel spacing for 433MHz band</td>
<td>100</td>
<td></td>
<td>kHz</td>
</tr>
<tr>
<td>$P_{RF10}$</td>
<td>Output power 10dBm setting</td>
<td>7</td>
<td>11</td>
<td>dBm</td>
</tr>
<tr>
<td>$I_{TX10dBm}$</td>
<td>Supply current @ 10dBm output power</td>
<td>30</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>$I_{RX}$</td>
<td>Supply current in receive mode</td>
<td>12.5</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>$RX_{SENS}$</td>
<td>Sensitivity at 0.1% BER</td>
<td>-100</td>
<td></td>
<td>dBm</td>
</tr>
</tbody>
</table>

Table A.1: Electrical specifications taken from the nRF905 transceiver’s datasheet
### Table A.2: Operation modes of the nRF905 transceiver taken from the device datasheet

<table>
<thead>
<tr>
<th>PWR_UP</th>
<th>TRX_CE</th>
<th>TX_EN</th>
<th>Operating Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Power down and SPI – programming</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Standby and SPI – programming</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>Read data from RX register</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Radio Enabled - ShockBurstTM RX</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Radio Enabled - ShockBurstTM TX</td>
</tr>
</tbody>
</table>

### Figure A.1: Pin layout of the nRF905 transceiver
### Table A.3: Pin functions of the nRF905 transceiver

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Pin function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TRX_CE</td>
<td>Digital input</td>
<td>Enables chip for receive and transmit</td>
</tr>
<tr>
<td>2</td>
<td>PWR_UP</td>
<td>Digital input</td>
<td>Power up chip</td>
</tr>
<tr>
<td>3</td>
<td>uPCLK</td>
<td>Clock output</td>
<td>Output clock, divided crystal oscillator full-swing clock</td>
</tr>
<tr>
<td>4</td>
<td>VDD</td>
<td>Power</td>
<td>Power supply (+3V DC)</td>
</tr>
<tr>
<td>5</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>6</td>
<td>CD</td>
<td>Digital output</td>
<td>Carrier Detect</td>
</tr>
<tr>
<td>7</td>
<td>AM</td>
<td>Digital output</td>
<td>Address Match</td>
</tr>
<tr>
<td>8</td>
<td>DR</td>
<td>Digital output</td>
<td>Receive and transmit Data Ready</td>
</tr>
<tr>
<td>9</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>10</td>
<td>MISO</td>
<td>SPI - interface</td>
<td>SPI output</td>
</tr>
<tr>
<td>11</td>
<td>MOSI</td>
<td>SPI - interface</td>
<td>SPI input</td>
</tr>
<tr>
<td>12</td>
<td>SCK</td>
<td>SPI - Clock</td>
<td>SPI clock</td>
</tr>
<tr>
<td>13</td>
<td>CSN</td>
<td>SPI - enable</td>
<td>SPI enable, active low</td>
</tr>
<tr>
<td>14</td>
<td>XC1</td>
<td>Analog Input</td>
<td>Crystal pin 1/External clock reference pin</td>
</tr>
<tr>
<td>15</td>
<td>XC2</td>
<td>Analog Output</td>
<td>Crystal pin 2</td>
</tr>
<tr>
<td>16</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>17</td>
<td>VDD</td>
<td>Power</td>
<td>Power supply (+3V DC)</td>
</tr>
<tr>
<td>18</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>19</td>
<td>VDD_PA</td>
<td>Power output</td>
<td>Positive supply (1.8V) to nRF905 power amplifier</td>
</tr>
<tr>
<td>20</td>
<td>ANT1</td>
<td>RF</td>
<td>Antenna interface 1</td>
</tr>
<tr>
<td>21</td>
<td>ANT2</td>
<td>RF</td>
<td>Antenna interface 2</td>
</tr>
<tr>
<td>22</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>23</td>
<td>IREF</td>
<td>Analog Input</td>
<td>Reference current</td>
</tr>
<tr>
<td>24</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>25</td>
<td>VDD</td>
<td>Power</td>
<td>Power supply (+3V DC)</td>
</tr>
<tr>
<td>26</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>27</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>28</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>29</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>30</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>31</td>
<td>DVDD_1V2</td>
<td>Power</td>
<td>Low voltage positive digital supply output for de-coupling</td>
</tr>
<tr>
<td>32</td>
<td>TX_EN</td>
<td>Digital input</td>
<td>TX.EN=&quot;1&quot;TX mode, TX.EN=&quot;0&quot;RX mode</td>
</tr>
</tbody>
</table>

*Table A.3: Pin functions of the nRF905 transceiver*
Figure A.2: Flowchart ShockBurst™ transmit of nRF905
Figure A.3: Flowchart ShockBurst™ receive of nRF905
A.2 Design information on the PIC24FJ128GA006

Figure A.4: Pin layout of the PIC24FJ128GA006 microcontroller
### Table A.4: DC characteristics of the PIC24FJ128GA006 taken from the device datasheet

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supply Voltage</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{DD}$</td>
<td>2.7</td>
<td>3.6</td>
<td>V</td>
<td>Regulator enabled</td>
</tr>
<tr>
<td>$V_{DD}$</td>
<td>$V_{DDCORE}$</td>
<td>3.6</td>
<td>V</td>
<td>Regulator disabled</td>
</tr>
<tr>
<td>$V_{DDCORE}$</td>
<td>2.0</td>
<td>2.75</td>
<td>V</td>
<td>Regulator disabled</td>
</tr>
<tr>
<td><strong>Operating Current</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{DD}$</td>
<td>20</td>
<td>32</td>
<td>mA</td>
<td>@+25C, 3.6V, 16MIPS</td>
</tr>
<tr>
<td>$I_{DD}$</td>
<td>20</td>
<td>32</td>
<td>mA</td>
<td>@+85C, 3.6V, 16MIPS</td>
</tr>
<tr>
<td><strong>Input Low Voltage</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O pins</td>
<td>VSS</td>
<td>0.2$V_{DD}$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>PMP pins</td>
<td>VSS</td>
<td>0.15$V_{DD}$</td>
<td>V</td>
<td>PMPTTL = 1</td>
</tr>
<tr>
<td>MCLR</td>
<td>$V_{SS}$</td>
<td>0.2$V_{DD}$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>OSC1 (HS mode)</td>
<td>$V_{SS}$</td>
<td>0.2$V_{DD}$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>SDAx, SCLx</td>
<td>$V_{SS}$</td>
<td>0.3$V_{DD}$</td>
<td>V</td>
<td>SMBus disabled</td>
</tr>
<tr>
<td>SDAx, SCLx</td>
<td>$V_{SS}$</td>
<td>0.8</td>
<td>V</td>
<td>SMBus enabled</td>
</tr>
<tr>
<td><strong>Input High Voltage</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O pins:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With Analog</td>
<td>0.8$V_{DD}$</td>
<td>$V_{DD}$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Digital-Only</td>
<td>0.8$V_{DD}$</td>
<td>5.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>MCLR</td>
<td>0.8$V_{DD}$</td>
<td>$V_{DD}$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>OSC1 (HS mode)</td>
<td>0.7$V_{DD}$</td>
<td>$V_{DD}$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>SDAx, SCLx</td>
<td>0.7$V_{DD}$</td>
<td>$V_{DD}$</td>
<td>V</td>
<td>SMBus disabled</td>
</tr>
<tr>
<td>SDAx, SCLx</td>
<td>2.1</td>
<td>$V_{DD}$</td>
<td>V</td>
<td>SMBus enabled</td>
</tr>
<tr>
<td><strong>Output Low Voltage</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O Ports</td>
<td>0.4</td>
<td>V</td>
<td>$I_{OL} = 8.5mA, V_{DD} = 3.6V$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>V</td>
<td>$I_{OL} = 6.0mA, V_{DD} = 2.0V$</td>
<td></td>
</tr>
<tr>
<td>OSC2/CLKO</td>
<td>0.4</td>
<td>V</td>
<td>$I_{OL} = 8.5mA, V_{DD} = 3.6V$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>V</td>
<td>$I_{OL} = 6.0mA, V_{DD} = 2.0V$</td>
<td></td>
</tr>
<tr>
<td><strong>Output High Voltage</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O Ports</td>
<td>2.4</td>
<td>V</td>
<td>$I_{OH} = -6.0mA, V_{DD} = 3.6V$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>V</td>
<td>$I_{OH} = -3.0mA, V_{DD} = 2.0V$</td>
<td></td>
</tr>
<tr>
<td>OSC2/CLKO</td>
<td>2.4</td>
<td>V</td>
<td>$I_{OH} = -6.0mA, V_{DD} = 3.6V$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>V</td>
<td>$I_{OH} = -3.0mA, V_{DD} = 2.0V$</td>
<td></td>
</tr>
</tbody>
</table>
Figure A.5: General block diagram of the PIC24FJ128GA microcontroller
A.3 Design information on the LM1117 voltage regulator

![Proposed connection circuit taken from the LM1117 datasheet](image)

*Required if the regulator is located far from the power supply filter.

Figure A.6: Proposed connection circuit taken from the LM1117 datasheet

A.4 Design information on the MAX3232 serial transceiver

![Proposed connection circuit taken from the MAX3232 datasheet](image)

Figure A.7: Proposed connection circuit taken from the MAX3232 datasheet
Appendix B

Multimedia guide

This thesis is accompanied by a DVD-ROM which is attached to the back page. A hierarchy of the files on the DVD is given in this Appendix for quick referencing.

B.1 Root directory

The root directory contains a:

- PDF copy of the thesis
- GUI executable with its accompanying Tables folder.
- List of the GPS coordinates of all the monitoring points.

B.2 Datasheets folder

This folder contains all the datasheets and additional design information of all the components used and considered during this thesis. These documents are subdivided into the following folders:

- RF link
- Microcontrollers
- Ethernet
- Serial interface
- Power supply
• Crystals
• Inductors
• Tantalum capacitors

### B.3 Flowcharts and pictures from the report folder

This folder contains all the pictures and figures used in the thesis, as well as some figures, pictures and saved results that did not feature in the thesis.

### B.4 MATLAB code folder

All the final implemented code and m-files designed during this study are supplied in this folder. The folder is subdivided into:

- Math Model - Contains the m-files of all the mathematical prediction models.
- Simulink Model - This folder contains the source files of various different models designed and used during this study.
- Network design - The MATLAB program created to analyze the network (refer to Chapter 2) is supplied in this folder, as well as the SRTM DEM files used during this study.

### B.5 Measured results folder

This folder contains the database containing all the measured results and some figures and images of the results.

### B.6 Other publications folder

This folder contains various publications used and referred to in this thesis.

### B.7 PCB design folder

The source files for the PCB design of the initial RSB, APB and the final RSB are supplied here.
B.8 Programming folder

This folder contains the source code of both the Embedded PIC software and the GUI software.

B.9 Software folder

Various freeware tools used during the duration of this study are supplied in this folder. They are:

- TeeChartOffice - A very handy database and spreadsheet plotting utility. The TeeChart component was also used in the GUI to provide the plotting features used in the Logs and Performance tabs.

- MPLAB IDE v7.6 and the MPLAB C30 compiler - These utilities were used to design the embedded code.

- Kashmir 3D - This is a mapping tool which enables a user to visualize DEM files in 3D.

- RFProp v110a - This small utility models the effect of an obstacle on radio propagation in a 2-dimensional environment. This program can be used to obtain a general feel of the importance of LOS links, by showing the effects caused by diffraction and refraction.

- Radio Mobile v8.5.8 - This full 3D radio propagation prediction software was used to analyze the network at hand. The full program with installation instructions is supplied in this folder.
Bibliography


