

**DEVELOPMENT & IMPLEMENTATION OF NEW
STRATEGIES FOR PARAMETRIC OPTIMIZATION OF
HYBRID COMMUNICATION NETWORK**

*A thesis submitted for the award of the
Degree of*

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

By

Ms. Sonal Jignesh Rane



**ELECTRICAL ENGINEERING DEPARTMENT
FACULTY OF TECHNOLOGY & ENGINEERING
THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA
VADODARA – 390 001
GUJARAT, INDIA**

March 2014

***Dedicated
To
My Family & Guide***

“The only source of knowledge is experience”

--Albert Einstein

Certificate

This is to certify that the thesis entitled, “**Development & Implementation of New Strategies for Parametric Optimization of Hybrid Communication Network**” submitted by **Ms. Sonal Jignesh Rane** in fulfillment of the degree of **DOCTOR OF PHILOSOPHY** in Electrical Engineering Department, Faculty of Technology & Engineering, The M. S. University of Baroda, Vadodara is a bonafide record of investigations carried out by her in the Department of Electrical Engineering, Faculty of Technology & Engineering, M. S. University of Baroda, Vadodara under my guidance and supervision. In my opinion the standards fulfilling the requirements of the Ph.D. Degree as prescribed in the regulations of the University has been attained.

March 2014

Prof. Satish K. Shah

Department of Electrical Engineering,
Faculty of Technology & Engineering,
The Maharaja Sayajirao University of
Baroda, Vadodara – 390 001

Head

Department of Electrical Engineering,
Faculty of Technology & Engineering,
The Maharaja Sayajirao University of
Baroda,
Vadodara – 390 001

Dean

Faculty of Technology & Engineering,
The Maharaja Sayajirao University of
Baroda, Vadodara – 390 001

Declaration

I, **Ms. Sonal Jignesh Rane** hereby declare that the work reported in this thesis entitled “**Development & Implementation of New Strategies for Parametric Optimization of Hybrid Communication Network**” submitted for the award of the degree of **DOCTOR OF PHILOSOPHY** in Electrical Engineering Department, Faculty of Technology & Engineering, The M. S. University of Baroda, Vadodara is original and has been carried out in the Department of Electrical Engineering, Faculty of Technology & Engineering, M. S. University of Baroda, Vadodara. I further declare that this thesis is not substantially the same as one, which has already been submitted in part or in full for the award of any degree or academic qualification of this University or any other Institution or examining body in India or abroad.

March 2014

Ms. Sonal Jignesh Rane

Acknowledgement

At the end of my thesis, it is a pleasant task to express my thanks to all those who contributed in many ways to the success of this study and made it an unforgettable experience for me.

First of all, I am extremely grateful to my research guide, **Prof. Satish K. Shah**, for his valuable guidance, scholarly inputs and consistent encouragement I received throughout my doctoral research work. I am also thankful to **Dr. S. K. Joshi**, Head of Electrical Engineering Department, Faculty of Technology and Engineering, The M.S.University of Baroda.

Words fail me to express my appreciation to **Dr. Dharmishtha Vishwakarma**, my best friend for her support. She was always beside me during the happy and hard moments to push me and motivate me whenever I couldn't able to find exact root. I can see the good shape of my thesis because of her help and suggestions in formatting the entire thesis.

Jignesh Rane, my husband, without whom this effort would have been worth nothing. His love, support and constant patience have taught me so much about sacrifice, discipline and compromise.

My parents **Ms. Jeshthaben Tank** and **Mr. Jaysukhbhai Tank**, my in-laws **Ms. Sunandaben Rane** and **Mr. Rameshbhai Rane** who have always supported, encouraged and believed in me, in all my endeavors and who so lovingly and unselfishly cared for **Dhairya**.

Dhairya Rane, my son, who was born before this dissertation was completed and who spent many days with relatives to allow me to focus. I am deeply sorry for the time we spent apart.

Jeshthaben Jaysukhbhai Tank, my mother and my sister **Charul Tank**, for looking after Dhairya at a moment's notice and for all their encouragement and profound understanding. My brother, **Umang J. Tank**, he supported me when sometimes I confused with technical knowledge.

I would like to thanks **Mr. Jayesh Bhatt**, my colleagues for their help and throughout support, teaching and non-teaching staff of Electrical Engineering Department, Faculty of Tech. & Engg., The M.S.University of Baroda for providing me co-operative environment.

Above all, I owe it all to **Almighty God** for granting me the wisdom, health and strength to undertake this research task and enabling me to its completion.

Sonal J. Rane

ABSTRACT

Hybrid Communication Networks have becoming more and more popular Networks. They provide both the advantages, wired and wireless communication. In these types of networks, Wireless Sensor Networks (WSNs) become very advancing and interesting topic in various applications.

WSN consists of battery operated portable devices which cause the limitations on capacity, bandwidth and computing power. Sustaining a route from a source to a destination may consume more bandwidth than is required to support the data traffic flow. Hence, packet routing techniques must be applied to provide long-range and large-scale communication in WSNs. Power consumption required by WSN Motes, topological features and traffic characteristics are the main constrains considered for analysis of WSN using simulation tools. In order to optimize the communication, it is important to know the characteristics of the traffic in advance.

WSN services are greatly supported by IEEE 802.15.4 Standard. This protocol grants exclusive use of a wireless channel for time-critical traffic load through a very attractive feature Guaranteed Time Slot (GTS) medium access control mechanism activated in the beacon-enabled mode based on the Superframe Structure. In this thesis, methodology for setting the relevant parameters of IEEE 802.15.4- compliant WSNs is analyzed and proposed. Considering the improvisations required in overcoming the flaw of traditional methods; Soft computing methods are used which differs from conventional (hard) computing in that, it is tolerant of imprecision, uncertainty, partial truth, and approximation. Here Traffic parameters of GTS Mechanism are optimized through soft computing techniques like Artificial Neural Network (ANN), Adaptive Neuro Fuzzy Inference System (ANFIS).

The effort took to examine the reliability of IEEE 802.15.4 OPNET simulation model through simulating wireless sensor network. In Hardware implementation, real WSN experiments were setup by using two MICAz nodes and one MIB520 gateway to evaluate the simulator's reliability. The results collected from MICAz test-bed experiments were compared with the results collected from the simulation experiments of the same scenarios. Also hardware implementation of soft computing algorithm is configured on Atlys Spartan 6 kit XC6SLX45 CSG324C FPGA platform using Xilinx ISE.

Table of Contents

List of Figures	V
List of Tables	XI
Acronyms	XI
1. Overview	1-8
2. Communication Network: An Introduction	9-20
2.1 <i>IEEE 802.3: Ethernet</i>	<i>10</i>
2.2 <i>IEEE 802.11: Wireless Lan</i>	<i>11</i>
2.3 <i>Hybrid Communication Networks</i>	<i>14</i>
2.4 <i>Wireless Sensor Networks</i>	<i>15</i>
<i>Summary</i>	<i>20</i>
3. WSN Simulators	21-39
3.1 <i>Simulation Platforms for WSN</i>	<i>21</i>
3.2 <i>The Characteristics of WSN Simulation</i>	<i>21</i>
3.2.1 <i>NS-2</i>	<i>23</i>
3.2.2 <i>GloMoSim</i>	<i>24</i>
3.2.3 <i>QualNet</i>	<i>24</i>
3.2.4 <i>TOSSIM</i>	<i>25</i>
3.2.5 <i>Avrora</i>	<i>27</i>
3.2.6 <i>OMNET++</i>	<i>27</i>
3.2.7 <i>SENS</i>	<i>28</i>
3.2.8 <i>MATLAB: TrueTime toolbox</i>	<i>28</i>
3.2.9 <i>OPNET</i>	<i>30</i>
3.2.9.1 <i>Specification Editors</i>	<i>31</i>
3.3 <i>Network Simulation using OPNET</i>	<i>32</i>
3.4 <i>Simulation Setup And Results</i>	<i>32</i>
3.4.1 <i>Wired and Wireless Network Comparison</i>	<i>33</i>
3.4.2 <i>Hybrid Network</i>	<i>37</i>
3.4.2.1 <i>Model Outline</i>	<i>37</i>

<i>Summary</i>	39
4. Development Support Tools	40-50
4.1 <i>MOTES</i>	41
4.2 <i>MATLAB</i>	42
4.3 <i>Simulink</i>	42
4.4 <i>Truetime Toolbox</i>	43
4.5 <i>Software Requirements</i>	45
4.6 <i>Compilation</i>	45
4.7 <i>SIMULATION SETUP</i>	46
4.7.1 <i>Process to run simulation model</i>	46
4.4.2 <i>Simulation Results</i>	48
<i>Summary</i>	50
5. IEEE 802.15.4 Structure	51-65
5.1 <i>Network Devices</i>	51
5.2 <i>Network Topologies</i>	52
5.3 <i>IEEE 802.15.4 Physical Layer (PHY)</i>	53
5.4 <i>IEEE 802.15.4 Medium Access Control</i>	53
5.5 <i>IEEE 802.15.4 Operational Modes</i>	54
5.6 <i>The Superframe Structure</i>	57
5.7 <i>Simulation Setup & Results</i>	59
<i>Summary</i>	65
6. IEEE 802.15.4 WSN: GTS Mechanism	66-82
6.1 <i>GTS Mechanism Evaluation</i>	66
6.2 <i>Related Work</i>	66
6.3 <i>The IEEE 802.15.4 Simulation Model</i>	67
6.4 <i>GTS Mechanism</i>	70
6.5 <i>Parameter Analysis</i>	73
6.5.1 <i>Throughput Analysis</i>	74
6.6 <i>Simulation Setup</i>	74
6.7 <i>Simulation Results And Discussion</i>	76
6.7.1 <i>Impact of Buffer size on GTS Throughput</i>	76
6.7.2 <i>Impact of Duty cycle on global statistics throughput.</i>	77

6.7.3	<i>Impact of Buffer size on Packet Medium Access Delay</i>	78
6.7.4	<i>Impact of the Superframe Orders on Wasted Bandwidth</i>	79
	<i>Summary</i>	82
7.	Soft Computing based WSN	83-98
7.1	<i>Soft Computing Models</i>	83
7.2	<i>Artificial neural networks</i>	83
7.2.1	<i>Basics of Artificial Neural Networks (ANN)</i>	85
7.2.2	<i>Types of ANN Learning</i>	85
7.2.3	<i>Back Propagation Network (BPN)</i>	86
7.3	<i>Adaptive Neuro Fuzzy Inference System (ANFIS)</i>	86
7.4	<i>Matlab Development tools</i>	87
7.4.1	<i>MATLAB</i>	87
7.4.2	<i>Simulink</i>	88
7.4.3	<i>Toolboxes</i>	89
7.5	<i>ANN based GTS Mechanism</i>	91
7.5.1	<i>Implementation of ANN GTS Mechanism</i>	92
7.6	<i>ANFIS based GTS Mechanism</i>	94
7.7	<i>Comparison of Traditional method, ANN, ANFIS</i>	95
7.8	<i>Soft GTS Mechanism Simulator</i>	96
	<i>Summary</i>	98
8.	Embedded Hardware: WSN	99-116
	SECTION A: Configuration of ANN on FPGA Kit	99
8.1	<i>Short introduction to FPGAs</i>	99
8.2	<i>FPGA design implementation of ANN</i>	100
8.3	<i>Design Implementation and Simulation Results</i>	101
8.4	<i>Test-bed Hardware Implementation</i>	103
	SECTION B: WSN Hardware implementation	105
8.5	<i>CROSSBOW MICA2 (MPR2400) MOTE Processor</i>	107
8.5.1	<i>CC2420 radio transceiver</i>	108
8.5.2	<i>MIB520 USB interface board</i>	108
8.5.3	<i>MDA100CA/MDA100CB</i>	109

8.5.4	<i>TinyOS</i>	109
8.6	<i>Software Description And Discussion</i>	110
8.6.1	<i>Software Development Tools</i>	110
8.6.2	<i>MOTE-VIEW Functionalities</i>	112
8.7	<i>Test-bed Hardware Setup and Implementation</i>	113
8.7.1	<i>Program Sensor Nodes</i>	113
8.7.2	<i>Experiment Results</i>	116
	<i>Summary</i>	116
9.	Conclusion & Future Scope	117-119
10.	Bibliography	120-135
	Appendix A: Development Environments	136-146
A.1	<i>Code Generation Using Xilinx ISE 14.6</i>	136
A.2	<i>Impact Setup</i>	139
A.3	<i>Atlys™ Board: Spartan-6 XC6SLX45 CSG324C</i>	142
A.4	<i>Programming WSN Motes</i>	142
	Appendix B: Photo gallery	147-150
	Appendix C: List of Programmed Files & Softwares	151
	Appendix D: User Interface	152-153
	Appendix E: List of Papers Based on Research work	154-156
E1:	<i>Publications - Proceedings/Referred (National/International) Journals</i>	154
E2:	<i>Presentations -Regional/National/International Conferences</i>	155
E3:	<i>Awards/Prizes</i>	156

List of Figures

Figure No.	Name of Figure	Page No.
2.1	Hybrid Communication Network	14
2.2	A taxonomy of current practice in wireless networking	15
3.1	TOSSIM Architecture	26
3.2	The TrueTime toolbox	29
3.3	Workflow	31
3.4(a)	Ethernet network model for 50 Ethernet stations	33
3.4(b)	Wireless network model for 50 users	33
3.5(a)	Traffic sent (bits/sec) of different scenarios having 25,50,100 users	34
3.5(b)	Traffic received (bits/sec) of different scenarios having 25, 50, 100 use	34
3.6	Collision count in wireless network	35
3.7	Throughput (bits/sec) of different scenarios on node 21 for Ethernet	35
3.8(a)	Data dropped for different scenarios for Wireless LAN	35
3.8(b)	Retransmission attempts of different scenarios for Wireless LAN	35
3.9	Throughput (bits/sec) of different scenarios for Wireless LAN	36
3.10(a)	Throughput of Ethernet and WLAN on node 07	36
3.10(b)	Throughput of Ethernet and WLAN on node 37	36
3.10(c)	Throughput of Ethernet and WLAN on node 97	36
3.11	Hybrid Network	37
3.12(a)	Light Traffic	38
3.12(b)	Heavy Traffic	38
3.13(a)	Delay for Light Traffic	38
3.13(b)	Delay for Heavy Traffic	38
4.1	WSN	40

4.2	Simulink model in truetype	47
4.3	Random Network Topology	47
4.4	Results on Command window for MOTES Simulink model	48
4.5	power (mW) V/S average visiting time (Sec)	48
4.6	Receiver threshold level (dbm) V/S Signal Reach	49
4.7	Power (dbm) V/S Signal Reach	49
4.8	Power (mW) V/S Average Visiting Priority for each node	49
4.9	Power (mW) V/S Average Visiting Priority	49
5.1	IEEE820.15.4/ZigBee protocol stack architecture	51
5.2	IEEE 802.15.4 Network Topologies	52
5.3	IEEE 802.15.4 Operational Modes	54
5.4(a)	Non Beacon mode	55
5.4(b)	Beacon mode	55
5.5	the Superframe Structure	56
5.6	Example of the structure of a Superframe	57
5.7	The Inter Framing Spacing	59
5.8	Number of Hops (Tree, Mesh)	61
5.9	End to end delay	61
5.10	End to end delay for Router 1	61
5.11	Load per PAN	61
5.12	Throughput	62
5.13(a)	Tree structure	62
5.13(b)	Mesh Route	62
5.14	Snapshot of the Network when Coordinator Fails	63
5.15	Load per PAN	64
5.16	PAN Affiliation	64

5.17	Global Output Report at simulation time 360, 60,720	64
6.1	The structure of the IEEE 802.15.4 Simulation Model	69
6.2	Packet flow structure in GTS enabled mode	71
6.3	The utilization of the transmission time inside the GTS	72
6.4	Simulation model	75
6.5(a)	GTS throughput v/s superframe order for 1 k buffer capacity	76
6.5(b)	GTS throughput v/s superframe order for 4 k buffer capacity	76
6.6	Global Throughput v/s BO	77
6.7(a)	PMAD v/s superframe order for 4 k buffer capacity	78
6.7(b)	PMAD v/s superframe order for 1 k buffer capacity	78
6.8	Wasted Bandwidth v/s superframe order for 4 k buffer capacity	79
6.9	Comparisons of Wasted Bandwidth Results	81
7.1	an Artificial Neuron	84
7.2	MATLAB Command Window	88
7.3	Simulink View	88
7.4	ANFIS Editor GUI	89
7.5	ANN with respective inputs and outputs	91
7.6	Training of Artificial Neural Network for Packet size	91
7.7(a)	Packet Medium Access Delay v/s Superframe Order for 700 bits Packet Size	92
7.7(b)	Packet Medium Access Delay v/s Superframe Order for 50 bits Packet Size	92
7.8 (a)	GTS Throughput v/s Superframe Order for 700 bits Packet Size	93
7.8(b)	GTS Throughput v/s Superframe Order for 50 bits Packet Size	93
7.9 (a)	Wasted B.W. v/s Superframe Order for 700 bits Packet Size	93
7.9 (b)	Wasted B.W. v/s Superframe Order for 50 bits Packet Size	93
7.10	ANFIS training	94

7.11	System Setup for ANFIS based GTS Mechanism	95
7.12(a)	Result analyses among traditional, ANN and ANFIS GTS Mechanism	95
7.12(b)	Result analyses among traditional, ANN and ANFIS GTS Mechanism	95
7.13	Soft GTS Mechanism Simulator	97
7.14	Snapshot of menus facility provided in given simulator	97
7.15	Snapshot of GTS Throughput result in designed simulator	98
8.1	Design Flow	99
8.2	Design implementation of ANN on FPGA	101
8.3	Output of ANN from ISIM Using VHDL code	102
8.4	Relative Difference	103
8.5	Hardware setup of ANN configuration on FPGA kit	104
8.6	Enlarge view of Spartan 6 demo Kit	104
8.7	Snapshot of programming FPGA kit	105
8.8	Output on LED for given combination of the inputs	105
8.9	Typical wireless sensor nodes size	106
8.10	Wireless sensor node components	106
8.11	MICAz mote [courtesy Crossbow]	107
8.12	Photo of top view of an MIB520CB	108
8.13	MDA 100CB	109
8.14(a)	Programming Environment of Motes	111
8.14(b)	Programming Environment of Motes	111
8.15	Snapshot of successful programming done in motes	112
8.16	Screenshot of the database in Health view	112
8.17	Experimental Test bed using MICAz Motes	113
8.18(a)	Simulation Test bed	114
8.18(b)	MICAz Nodes & Gateway Setup	114
8.19	Snapshot of connecting mote on gateway through USB port	115

8.20	Snapshot of Uploading program	115
8.21	Test bed versus Simulation results	116
A.1	Project Navigator Desktop Icon	136
A.2	New Project Wizard—Create New Project Page	136
A.3	New Project Wizard—Device Properties Page	137
A.4	Adding VHDL Test Bench	137
A.5	Process Pane	138
A.6	ISim Properties Dialog Box	138
A.7	ISim Graphical User Interface	139
A.8	Cable Set up	140
A.9	Cable Communication Setup	140
A.10	Snapshot of Boundary Scan	141
A.11	Program Launch Window	141
A.12	Snapshot of Programming Notepad 2	143
A.13	Output Section of PN2	143
A.14	Mode Configurations	144
A.15	Gateway Configuration	145
A.16	Database Configurations	145
A.17	Sensor Board Configurations	146
A.18	MICAz Nodes & Gateway Setup	146
B.1	Spartan-6 XC6SLX45 CSG324C Evaluation Board	147
B.2	System Setup for FPGA Implementation of ANN	147
B.3	Output observed on LEDs	148
B.4	Crossbow's real time Hardware foe WSN	148
B.5	Mote connected on Gateway Setup	149
B.6	Hardware Setup	149
B.7	Status of Mote when data received	150

B.8	Snapshot of uploading Program	150
D.1	IEEE 802.15.4 WSN -User Interface	152
D.2	IEEE 802.15.4 WSN -Documentation	152
D.3	IEEE 802.15.4 WSN –Hardware Implementation Menu	153
D.4	IEEE 802.15.4 WSN –Demo	153

List of Tables

Table No.	Name of Table	Page No.
2.1	Industrial, Scientific and Medical (ISM) bands	12
3.1	Characteristics of WSN Simulations	22
3.2	Traffic Specification	38
4.1	Simulation parameters	47
6.1	GTS setting	73
6.2	GTS Traffic Parameters	73
6.3	SIMULATION PARAMETERS	75
6.4	Simulation Parameters and resultant parameter	81
8.1	Comparison of ANN results of MATLAB and FPGA	102

Acronyms

MEMS	Micro Electro Mechanical Systems
HCN	Hybrid Communication Networks
QoS	Quality Of Service
WSN	Wireless Sensor Networks
WANET	Wireless Ad-Hoc Network
LR-WPANs	Low-Rate Wireless Personal Area Networks
MAC	Medium Access Control
ADSL:	Asymmetric Digital Subscriber Line
VDSL	Very-High-Bit-Rate Digital Subscriber Line
LAN	Local Area Network
FDDI	Fibre Distributed Data Interface
CSMA/CD	Carrier Sense, Multiple Access With Collision Detect
Wi-Fi	Wireless Fidelity
LLC	Logical Link Control
RTS	Request To Send
CTS	Clear To Send
ISM	Industrial, Scientific And Medical

AP	Access Point
BSS	Basic Service Set
HCN	Hybrid Communication Network
NS-2	Network Simulator-2
TCP	Transport Control Protocol
GloMoSim	Global Mobile Information System Simulator
QualNet	Quality Networks
TOSSIM	Tinyos Mote Simulator
OMNeT	Objective Module Network Test-Bed In C++
SENSIM	Sensorsimulator
OPNET	Optimised Network Engineering Tools
FSMs	Finite State Machines
ICI	Interface Control Information
PDFs	Probability Density Functions
SNR	Signal To Noise Ratio
GUI	Graphical User Interface
IEEE	Institute Of Electrical And Electronics Engineers
WPAN	Wireless Personal Area Network.
FFD	Full Function Device
RFD	Reduced Function Device
PAN	Personal Area Network
ED	End Device
C	Coordinator
R	Router
PHY	Physical Layer
PPDU	Physical Protocol Data Units
DSSS	Direct Sequence Spread Spectrum
CSMA/CA	Carrier Sense Multiple Access / Contention Avoidance
ETE	End To End Delay
GTS	Guaranteed Time Slot
PMAD	Packet Medium Access Delay
SO	Superframe Order
BO	Beacon Order
SD	Superframe Duration
BI	Beacon Interval
TS	Time Slot duration

QPSK	Quadrature Phase Shift Keying
CAP	Contention Access Period
CFP	Contention Free Period
IFS	Inter Frame Spacing
ANN	Artificial Neural Network
ANFIS	Adaptive Neuro Fuzzy Inference System
SC	Soft Computing
FPGAs	Field Programmable Gate Arrays
PLBs	Programmable Logic Blocks
VHSIC	Very High Speed Integrated Circuit
HDL	Hardware Description Languages
RISC	Reduced Instruction Set Computer
MPR	Mote Processor Radio
MIB	Mote Interface Boards
RSSI	Received Signal Strength Indicator
ISP	In-System Processor
MLP	Multilayer Perceptron



Chapter 1



Overview



This chapter gives brief introduction of the work considered for thesis. Here, brief overview of the contents of the chapters is described.

The development of microelectronics began in 1948 and continued with the miniaturization of sensors during last ten years. Today, Microsystems which are used in silicon micro technology are called Micro Electro Mechanical Systems (MEMS). Micro sensors are defined as very small sized devices that convert humidity, moisture, temperature, pressure, pollutants, light intensity, etc. into electrical signal. Now, Integration of MEMS with the radio frequency wireless communication technique will put forward low power consuming, cheap and communicating small devices, called Wireless Sensor Networks (WSN). WSN is the mixture of wired and wireless communication and combination of both the communications are known as Hybrid Communication Networks (HCN). HCN have spatially distributed transceivers with micro sensors that give the possibility of monitoring and collecting data easily and quickly. Nowadays, these devices are used in many different application areas like home automation, forest fire monitoring, traffic control, and noise & pollutant detection in the crowded areas.

As networks become more and more complicated and applications more and more demanding, a very common network topology for state-of-the-art multimedia applications is a hybrid wired/wireless architecture. The impact of this topology's particularities, like different throughput and bandwidth or packet format, on the Quality of Service (QoS) demanding nature of multimedia applications, is a growing research field.

HCN is one with both wired and wireless connections. Because in most cases, a transceiver-equipped PC or other device known as an access point is used and connected to a wired network, such as the telephone network or a wired LAN, which uses some type of standard cabling. This access point can receive and transmit data between the wireless and wired worlds. The chief advantage of a wireless network is mobility and flexibility. Other than that, both wired and wireless networks are equally easy (or difficult) to set up, depending on the organization's size and complexity.

The IEEE 802.15.4 [1] protocol has recently been adopted as a communication standard for low data rate, low power consumption and low cost Wireless Sensor Networks (WSNs). This protocol is quite flexible for a wide range of applications by adequately tuning its parameters and it also provides real time guarantees by using the Guaranteed Time Slot (GTS) mechanism. This feature is quite attractive for time-sensitive WSN applications.

WSN is a wireless network consisting of spatially distributed devices integrated with sensors to cooperatively monitor physical or environmental conditions such as temperature, pressure, humidity, vibration, motion or pollutants using wired network, and exchange the information at different locations using wireless networking. Each wireless device is also called a node that behaves individually. Each node has one or more sensors integrated on it. In addition to these sensors, a node is also equipped with a transmitter and a receiver. These transmitter and receiver are used for wireless communications with other nodes or directly with the gateway. The gateway is responsible for transmitting sensor data from the sensor patch to the remote base station that provides Wireless Ad-Hoc Network (WANET) connectivity and data logging through a local transit network. Finally, the data is available to researchers through a user interface. The other parts of a sensor node are the microcontroller and the battery (as the energy source). Battery-powered embedded systems, such as WSN motes, require low energy usage to extend system lifetime. WSN motes must power sensors, a processor, and a radio for wireless communication over long periods of time, and are therefore particularly sensitive to energy use. Recent techniques for reducing WSN energy consumption, such as aggregation, require additional computation to reduce the cost of sending data by minimizing radio data transmissions. Larger demands on the processor will require more computational energy, but traditional energy reduction approaches, such as multi-core scaling with reduced frequency and voltage may prove heavy handed and ineffective for motes. Battery-powered embedded systems carefully manage energy consumption to maximize system lifetime. WSNs, made up of many “mote” devices, and are often designed to operate for months without intervention.

Time-critical applications for wireless sensor networks (WSNs) are an important class of services supported by the standard IEEE 802.15.4. Understanding the delay in the packet delivery is fundamental to assess performance limitation for the standard. The IEEE 802.15.4 standard specifies the physical layer and MAC sub-layer for Low-Rate Wireless Personal Area Networks (LR-WPANs). The ZigBee standard is closely associated with the IEEE 802.15.4 protocol and specifies the network (including security services) and application (including objects and profiles) layers. The IEEE 802.15.4 Task Group (TG4) [1], together with the ZigBee Alliance [2], has developed an entire communication protocol stack for LR-WPAN [3].

When an IEEE 802.15.4-compliant WPAN disables the generation of periodic beacon frames i.e. non-beacon enabled mode, all nodes in the network compete to gain

access to the medium using non-slotted CSMA/CA. The performance of the IEEE 802.15.4 protocol has been subject to few research studies, focusing more on the performance of its CSMA/CA protocol, is discussed in [4-6] or its general characteristics using simulations [7]. The advantage of the non-beacon enabled mode, with regards to WSN application requirements, is that it easily allows scalability and self-organization. However, the non-beacon enabled mode does not provide any guarantee to deliver data frames, specifically within a certain deadline. For time-critical applications, real time guarantees may be achieved with the beacon-enabled mode. This mode offers the possibility of allocating/deallocating time slots in a superframe, called Guaranteed Time Slots (GTSs) [8-12] and providing predictable minimum service guarantees. Having a minimum service guarantee, it is possible to predict the worst-case real-time performance of the network.

The IEEE 802.15.4 standard grants exclusive use of a wireless channel for time-critical traffic through GTS medium access control (MAC) mechanism. The GTS is activated in the beacon-enabled mode based on the superframe structure. A node has to use at least one whole GTS in a transmission. However, each node may not fully utilize its transmission capacity in a particular time slot if the packet arrival rate is too small. Hence, bandwidth utilization is reduced. Queue management and buffer dimensioning at the nodes becomes an important factor to quantify the optimal parameters to tune the network for a better performance without making any modifications to the existing protocol [13].

Soft Computing Techniques (Artificial Neural Networks, Genetic Algorithms and Fuzzy Logic Models) have been recognized as attractive alternatives to the standard, well-established “hard computing” paradigms. Traditional hard computing methods are often too cumbersome for today’s problems. They always require a precisely stated analytical model and often a lot of computational time. Soft computing techniques, which emphasize gains in understanding system behavior in exchange for unnecessary precision, have proved to be important practical tools for many contemporary problems.

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the connections between elements largely determine the network function. You can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements. Typically, neural networks are adjusted, or trained, so that a particular input leads to a specific target output. ANN can be used for complex relationships

between inputs and outputs. ANN in most cases is adaptive systems that change their structure based on external or internal information flowing through them and use a connectionist approach to process information [14]. Back propagation feed forward networks are standard neural networks for any supervised learning pattern recognition.

The ANFIS [15] was proposed many years ago and is widely used in research works. The ANFIS reveals an efficient learning network and its applications can be found in many works in the literature [16-18]. The acronym ANFIS derives its name from adaptive Neuro-Fuzzy inference system. Using a given input/output data set, the toolbox function ANFIS constructs a fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using either a back propagation algorithm alone or in combination with a least squares type of method. This adjustment allows your fuzzy systems to learn from the data they are modeling.

This thesis provides brief idea about IEEE 802.15.4, whose MAC protocol supports two operational modes that can be selected by a central controller of the Personal Area Network (PAN), called PAN Coordinator, viz., Beacon-enabled mode, Non Beacon-enabled mode. Both modes are examined, evaluates GTS mechanism in beacon-enabled mode for WSN based on IEEE 802.15.4 standard and optimize based on given set of parameters.

The proposed mechanism is used to carry out a set of experiments and to compare the obtained simulation results with the ones that were previously obtained using an OPNET simulation model based on Network Calculus. An accurate OPNET simulation model of the IEEE 802.15.4/ZigBee protocols [19] focusing on the implementation of the Guaranteed Time Slot (GTS) mechanism and ZigBee hierarchical routing strategy is discussed. Optimization of GTS mechanism is one of the recent issue when we use WSN in beacon enabled mode and working on Contention Free Period (CFP).

This has inspired the author to work in this direction of development of new strategies for performance of the network using soft computing techniques such as Fuzzy Logic, Fuzzy models, ANN, ANFIS, Genetic algorithm. When exact mathematical model is not possible, soft computing technique based methods can be develop and implement on computational platforms. The use of software development support tools such as MATLAB, TRUE TIME Toolbox and use of OPNET [20] simulator simplifies simulation, implementation and testing of WSN and hardware for the optimization techniques.

There are several implementations of the IEEE 802.15.4/ZigBee protocols supported by different hardware platforms [21-31]. These were developed in C language and programmed directly in the microcontroller without any supporting operating system (like TinyOS). Additionally these implementations only support the non-beacon enabled mode, therefore allowing the construction of ZigBee standard mesh networks, but not of beacon-enabled Star and Cluster-Tree networks [19]. The experimental test bed is based on an IEEE 802.15.4 star network operating in a beacon-enabled mode, with PAN Coordinator and end devices.

Research Objectives and Contributions

- ✂ Development of the model for wired and wireless network and simulation of the model on software tool focusing on the parameters viz., collision count, traffic received, delay, throughput, data dropped, media access delay.
- ✂ Survey of topologies for IEEE 802.15.4 and routing protocols.
- ✂ Simulation study of routing protocols and topology on performance of IEEE 802.15.4.
- ✂ Development and evaluation for IEEE 802.15.4 and Zigbee protocols focusing on the Routing protocol and Zigbee hierarchical routing strategy in non-beacon enabled Wireless Sensor Networks (WSN).
- ✂ Selection of routing protocol and strategy for non-beacon enabled WSN employing IEEE 802.15.4.
- ✂ Study and analysis of performance of motes employing MATLAB Simulink and Truetime.
- ✂ Study of superframe structure and various mechanisms for parametric optimization and proposing new strategy of parametric optimization.
- ✂ Survey of Preliminary Technique and proposing new strategy of Parametric Optimization for WSN.
- ✂ Design, implementation and evaluation of an accurate simulation model for IEEE 802.15.4 protocol focusing on the Guaranteed Time Slot (GTS) mechanism in beacon-enabled, star topology in Wireless Sensor Networks (WSNs).
- ✂ The formulation, implementation and evaluation of soft computing methods to optimize the parameters of GTS Mechanism.
- ✂ Development of SOFT computing algorithms using ANN, ANFIS.

- ✱ Design of SOFT GTS mechanism with user interface employing matlab.
- ✱ Implementing WSN on embedded hardware.

The thesis is organized in the form of ten chapters as follows:

Chapter: 1

The chapter provides an overview and the context for the remainder of the thesis.

Chapter: 2

This chapter gives brief idea about Hybrid Network. It describes an overview and classification of networks used for communication. A survey of different networks viz., wired and wireless networks, different wireless ad hoc network like mobile ad hoc network, vehicular ad hoc network, wireless sensor network, wireless mesh network is described.

Chapter: 3

This chapter describes the study of different network simulators like NETWORK SIMULATOR NS-2, OPNET, OMNET++, GLOMOSIM, and QUALNET. Performance of wireless and wired networks as well as comparison is evaluated using OPNET simulation tool. For wired network, collision count, traffic received, delay, throughput is studied while for wireless network, data dropped, traffic received, media access delay, and throughput is studied. For comparison of both wired and wireless networks, the performance parameters throughput is investigated [32]. All these performance is carried out by varying number of users.

Chapter: 4

In this chapter, study of motes behavior using True time simulator, MATLAB based networked Simulator is discussed. Truetime is a MATLAB/Simulink-based simulator for networked and embedded control systems. The simulator software consists of a Simulink block library and a collection of MEX files. The kernel block simulates a real-time kernel executing user-defined tasks and interrupt handlers. The various network blocks allow nodes (kernel blocks) to communicate over simulated wired or wireless networks [33]. In [34] Wireless Sensor Network is simulated using Truetime toolbox. In this chapter, behavior of wireless motes has been examined with respect to parameters like Transmission Power, Receiving threshold power, Average visiting time [35].

Chapter: 5

This Chapter provides an overview of the most relevant aspects of the IEEE 802.15.4 and ZigBee protocols. Design of simulation model with respect to the specifications of IEEE 802.15.4 standard for WSN is discussed. The primary goal of routing protocols [36] which are designed for WSNs is to maintain energy efficient and reliable paths between different nodes in the network without generating a lot of overhead. The chapter discusses simulation and evaluation of two scenarios, where we examined the topological features and performance of the IEEE 802.15.4 standard using OPNET simulator [37]. The comparative results for two topologies are reported for the performance metrics like: Number of hops, End to End Delay and Load of network.

Chapter: 6

Timeliness is an important feature of the IEEE 802.15.4 protocol, turning it quite appealing for applications under timing constraints. Because of this attractive feature, it is used in real time for time constraint data delivery which is provided by Guaranteed Time slot mechanism. This chapter explores the underutilization of bandwidth in WSN and analyses GTS mechanism by evaluating throughput in OPNET Modeler [38].

Chapter: 7

This Chapter gives a brief overview and describes theoretical background of the soft computing techniques such as ANN, ANFIS. The most popular tools used by the researchers for development and simulation study of the system under test such as MATLAB and associated tool boxes for software development and testing are also described. Toolboxes available for deploying soft computing techniques in MATLAB and used in our research work for the design and testing of proposed techniques are described in detail. In this chapter, GTS mechanism is examined using proposed methods based on soft computing techniques such as ANN and ANFIS. The implementation of algorithm in MATLAB is discussed. Training of ANN [39] and ANFIS techniques are discussed and compare with the results obtained from traditional method (OPNET simulation). The performance improvement of GTS mechanism is also discussed.

Chapter: 8

This chapter describes the hardware implementation of WSN. In beacon enabled mode, sensor nodes (motes) supporting the IEEE 802.15.4 Standard can be used as PAN coordinator, end device forming star topology. Our GTS mechanism based experiments use the Embedded Hardware for Hardware implementation.

Chapter: 9

Conclusion & Future Scope: Final conclusions and future extensions of the work and future scope in this field are elaborated in this chapter.

Chapter: 10

Thesis ends with Bibliography which includes the list of references used in each chapter and list of publication and presentations done based on this work.



Chapter 2



Communication Network: An Introduction



This Chapter describes the brief about networks available for communication. This chapter gives brief idea about Hybrid Network. It describes an overview and classification of networks used for communication. A survey of different types of networks viz., wired and wireless networks, different wireless ad hoc network like wireless sensor network, Wireless local area Network is described.

Today, communication enters our daily lives in so many different ways that it is very easy to overlook the multitude of its facts. The telephone at our hands, the radios and televisions in our living rooms, the computer terminals in our offices and homes, and our newspapers are all capable of providing rapid communications from every corner of the globe [1].

A computer network communication has following properties:

- ✖ Facilitates interpersonal communications
 - People can communicate efficiently and easily via email, instant messaging, chat rooms, telephone, video telephone calls, and video conferencing.
- ✖ Allows sharing of files, data, and other types of information
 - Authorized users may access information stored on other computers on the network. Providing access to information on shared storage devices is an important feature of many networks.
- ✖ Allows sharing of network and computing resources
 - Users may access and use resources provided by devices on the network, such as printing a document on a shared network printer. Distributed computing uses computing resources across a network to accomplish tasks.
- ✖ May be insecure
 - A computer network may be used by computer Hackers to deploy computer viruses or computer worms on devices connected to the network, or to prevent these devices from accessing the network (denial of service).
- ✖ May interfere with other technologies
 - Power line communication strongly disturbs certain forms of radio communication, e.g., amateur radio. It may also interfere with last mile access technologies such as ADSL and VDSL.
- ✖ May be difficult to set up

- A complex computer network may be difficult to set up. It may be costly to set up an effective computer network in a large organization.

With the recent developments in the communication technology, the networking of more devices is possible. The communication networks used to connect devices to form a computer network include electrical cable, optical fiber and radio waves.

A widely-adopted family of communication network used in local area network (LAN) technology is collectively known as Ethernet. The media and protocol standards that enable communication between networked devices over Ethernet are defined by IEEE 802. Ethernet encompasses both wired and wireless LAN technologies. Wired LAN devices transmit signals over cable media. Wireless LAN devices use radio waves or infrared signals as a transmission medium.

2.1 IEEE 802.3: Ethernet

Wired Local Area Networking [1] includes several technologies such as Ethernet [2], token Ring, Token bus, FDDI (Fibre distributed data interface) and ATM (asynchronous transfer mode) LAN (local area networks) [3]. Some of these technologies survived for a while, but Ethernet is by far the dominant technology. Evolution from a 10Mbps Standard Ethernet to bridged Ethernet and then to a switched Ethernet paved a way for faster Ethernet. IEEE 802.3 Standard specifies Carrier Sense, Multiple Access with Collision Detect (CSMA/CD) as the access method for first-generation 10-Mbps Ethernet, a protocol that helps devices share the bandwidth evenly without having the two devices transmit at the same time on the network medium. The Ethernet is a working example of the more general CSMA/CD local area network technology.

The Ethernet is a multiple-access network, meaning that a set of nodes sends and receives frames over a shared link. When the two devices transmit at the same time the collision can occur. This collision generates a jam signal that causes all nodes on the segment to stop sending data, which informs all the devices that a collision has occurred. The carrier sense in CSMA/CD means that all the nodes can distinguish between an idle and a busy link. The collision detect means that a node listens as it transmits and can therefore detect when a frame it is transmitting has interfered (collided) with a frame transmitted by another node. The Ethernet is said to be a 1-persistent protocol because an adaptor with a frame to send transmits with probability 1 whenever a busy line goes idle.

This Carrier Sense, Multiple Access with Collision Detect (CSMA/CD) protocol was created to overcome the problem of collisions that occur when the packets are transmitted simultaneously from different nodes.

Even though the wired reduces the cost of cabling and gives more flexibility and re-configurability, it is not fully configurable. It includes the cost of cabling damage to the cables temporary abandon of the work progress due to the reconfiguration of the entire system. When operating under the harsh conditions this is not essential and could also lead to the damage of the infrastructure. Suppose if there are any mobile parts then the wires should also be moving a very difficult constraint to meet without any issues. This led to the induction of wireless technologies in industrial networks.

2.2 IEEE 802.11: Wireless LAN

Wireless local area networks (WLANs) [4] extend the boundaries of traditional wired local area networks (LANs) by unleashing the constrained flow of wire-line data to saturate the surrounding area. Wireless communication offers significant advantages to both users and network designers. Users gain flexible mobility to work anywhere within radio communication range of a network access point and seamlessly retrieve network resources. Roaming around the network, pervasive devices discover each other and permit users to benefit from context aware applications. Network designers gain tremendous advantages in rapid network building, upgrading, and reconfiguration. IEEE 802.11 is a recent standard developed for wireless local area networks (WLANs). IEEE 802.11 is a multiple access protocol in which stations in the network must compete for access to the shared communications medium to transmit data. IEEE 802.11 uses a carrier sensing capability to determine if the communications medium is currently being used [3]. If two or more stations in the network transmit at the same time (i.e., a collision occurs), stations retransmit their data after random periods of time as in Ethernet. Wi-Fi (Wireless Fidelity) Technology, referred as the IEEE 802.11 communications standard for WLAN, is the popular wireless networking technology that uses radio waves to provide wireless high-speed Internet and network connections. The IEEE 802.11 data link layer is divided in two sub layers: Logical Link Control (LLC) and Media Access Control (MAC). LLC is the same as in 802 LAN allowing for very simple bridging from wireless to wire networks. MAC is different to WLANs. The first method in MAC is CSMA with collision avoidance protocol. This protocol is to ask each station to listen before action. If

the channel is busy, the station has to wait until channel is free. Another method in MAC is called RTS/CTS to solve hidden-Node problem [5].

Wireless communications also offer significant network challenges [6]. Since the broadcast medium is shared by many devices and networks, channel controls must be implemented at both the network and station level to facilitate fair, regulated access to the medium. The Federal Communications Commission [7] regulations allocated the three Industrial, Scientific, and Medical (ISM) bands shown in Table 2.1 for unlicensed use under strict power guidelines to prevent interference.

Band	Frequency Range
UHF ISM band	902 to 928MHz
S-band ISM	2.4 GHz to 2.5 GHz
C-band ISM	5.725 to 5.875 GHz

Table 2.1 Industrial, Scientific and Medical (ISM) bands

Each band is subdivided into channels with much lower throughput capacity than wired channels. Additionally, the wireless environment introduces significant path loss uncertainty, constantly changing in both the space and time domain. Finally, along with the freedom of open wireless network boundaries, the inability to control both active and passive access to the medium increases network security vulnerabilities.

WLAN networks are implemented using IEEE 802.11 standard compliant devices and are classified as either infrastructure or ad hoc networks. Stations in an infrastructure WLAN network transfer data to a central access point (AP). The AP either forwards the message into a distribution system or relays the message to another station within the AP's basic service set (BSS), all stations associated to a particular AP. Ad hoc WLAN network stations form a peer-to-peer relationship with nodes within communications range to form an independent BSS (IBSS).

Although the deployment of wireless technologies completely eradicates the problems of the wired Networks, it is not possible to replace all the systems with wireless technology because they are highly prone to errors. Real time Communication, in which timely delivery of data is important, the missing of deadlines due to network traffic can affect the system performance. The Real time system applications mostly demands lesser bandwidth, long distance communication and more channels for multi device communication. This can be achieved with the help of wireless technology.

As wired LANs already exist since they are older and because wired LANs allow for more resources because they generally have higher rates and are not limited in terms of size and power consumption as wireless LANs, there has been an increasing demand to connect wireless LANs to wired LANs [8].

It was also noticed that the needs for interconnecting wireless LANs to an existing wired back-bone LAN is increasingly becoming essential in many areas. So far we have discussed that Networks can be classified according to the nature of the medium of their links, to wired and wireless. As the complexity degree of these applications increases over the years, different network traffic must be handled and carried over different mediums in a unique homogeneous way. Hence, the need for interoperability in heterogeneous networks with hybrid structure is undoubtedly a major requirement, when integrating communication scenarios for home and industrial applications.

This type of hybrid network is becoming essential, as it allows for resource sharing and data transfer between the new wireless networks and the already established wired networks.

As IEEE 802.11 [9] and Ethernet [10] are becoming the most common and widely used WLAN/LAN standards, an interoperable architecture is required in order for communication to be treated transparently at the higher-levels. However, it has to be pointed out that from the user point of view, the entire system is seen as a black box and is expected to function equally well, independently from network heterogeneity. With the growing industry needs and expansion of the physical setups the conventional point-to-point technique does not meet the real time requirements such as modularity, decentralization of control, integrated diagnostics, quick and easy maintenance, and low cost because of the number of cables being proportional the square of the devices and other problems [11]. Therefore, the protocol can be redefined and permits to integrate both wired and wireless technologies to meet the industry requirement without affecting the stability of the plant. On the other hand, when dealing with hybrid wired/wireless networks, questions arise regarding QoS and power awareness issues especially concerning the wireless part of the hybrid network.

Theoretically speaking, all types of network topologies used for wired LANs can be used for Wireless LANs, but practically this is not true. The fact that wireless communication channels have different characteristics than wired channels, and the demand for mobility and ad-hoc connectivity in WLANs are the reasons why this is not true [12,13].

2.3 Hybrid Communication Networks

Hybrid Communication Network (HCN) (Figure 2.1) is one with both wired and wireless connections. Because in most cases, a transceiver-equipped PC or other device known as an access point is used and connected to a wired network, such as the telephone network or a wired LAN, which uses some type of standard cabling. This access point can receive and transmit data between the wireless and wired worlds. The chief advantage of a wireless network is mobility and flexibility. Other than that, both wired and wireless networks are equally easy (or difficult) to set up, depending on the organization's size and complexity.

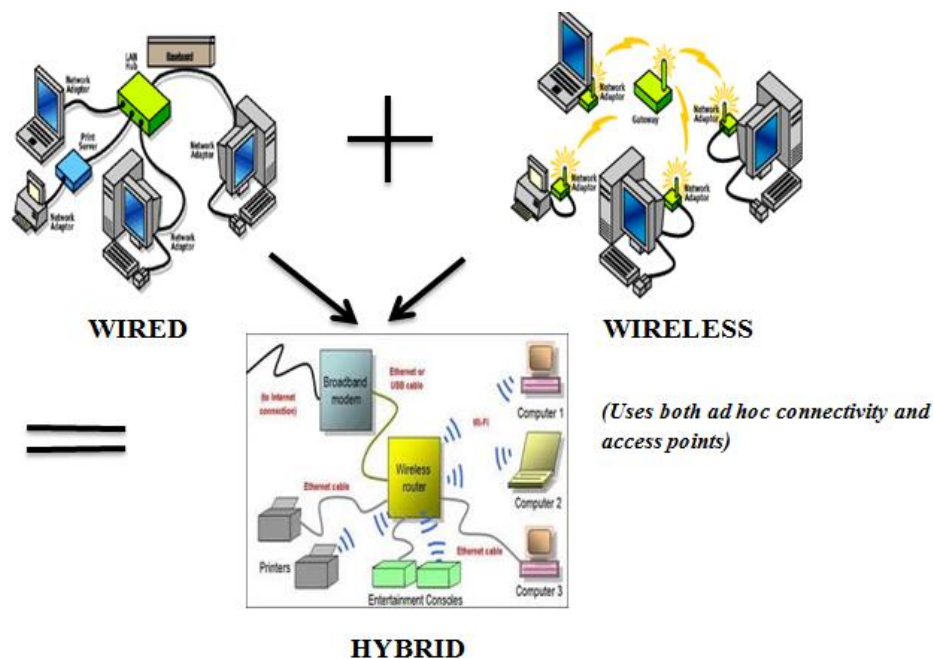


Figure 2.1: Hybrid Communication Network

Classification of Different wireless networks can be done on following basis:

- ✖ infrastructure,
- ✖ mobility
- ✖ size of network.

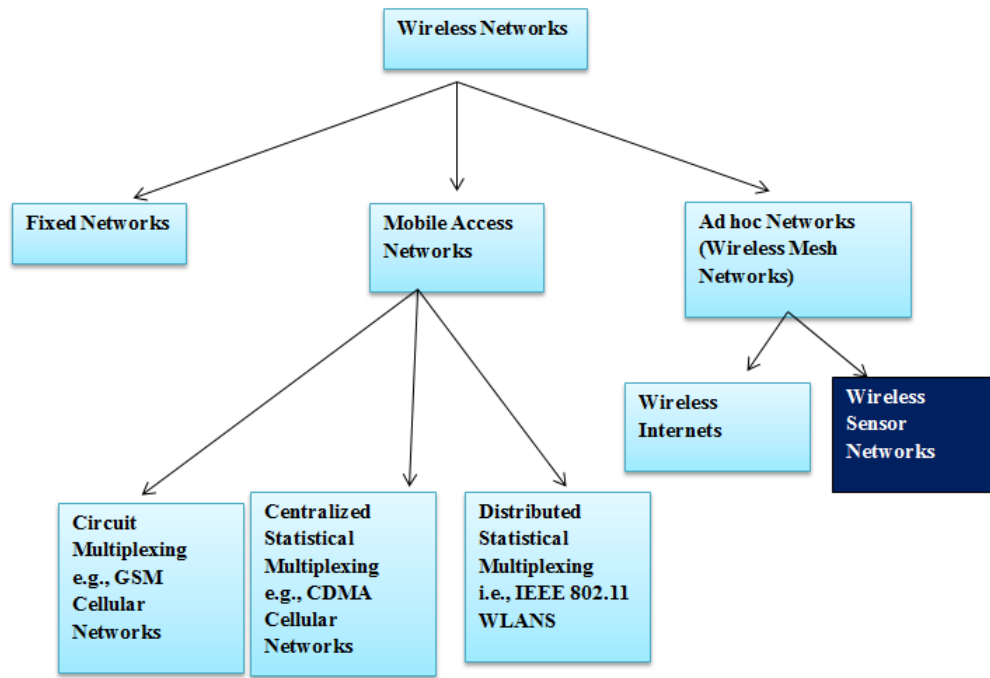


Figure 2.2 : A taxonomy of current practice in wireless networking [13]

Figure 2.2 depicts classification of the taxonomy of current most popular networks used for communication. Among them, Wireless Sensor Network is one of the most useful in many real time Applications.

2.4 Wireless Sensor Networks

Wireless sensor networks (WSN) may consist of several to thousands of homogeneous or heterogeneous sensors that share the need to organize for data collaboration or network data collection sink routing [14]. Small system platforms which integrate sensors, processors, and transceivers are referred to as motes. Remote sensing platforms are typically characterized by reduced processing capabilities, limited memory capacities, and fixed battery supplies. The WSN energy consumption falls into three categories: sensing, computing, and communicating. Analysis conducted by Soharabi et al. [15] demonstrates that the communications costs dominate a WSN sensor platform's power budget. WLAN networks were designed to minimize delay and maximize throughput, but they do not provide the energy efficiency demanded by WSN networks. As technology makes the hardware smaller, WSN research continues developing innovative, energy-saving techniques at all network protocol layers in order to engineer sensor platforms which can operate unattended for months or even years. The WSN networks must also be scalable to support extremely dense sensor fields. Applications for energy-efficient WSN networks include homeland defense nuclear/biological/chemical

(NBC) sensing, military surveillance, and environmental sensing [16-18]. These applications generally work in a self-organizing, clustered environment that supports either a single application or collaborative applications. WSN network design requires trade-offs in throughput and latency to extend network lifetimes.

Given such a diverse set of applications and requirements [19], it should come as no surprise that the constraints which guide the design and deployment of wireless sensor networks differ, sometimes substantially, from those that hold in wireless ad hoc networks [20]. Following are the constraints, discussed in more detail.

⌘ **ENERGY EFFICIENCY:** Probably the most important difference is due to the fact that sensor nodes typically operate on limited battery power, which means that the maximization of network lifetime (and, consequently, minimization of power consumption) is a sine qua non for sensor networks. On the contrary, power consumption is seldom the critical requirement for ad hoc networks.

According to [21], the constraint of minimal energy consumption translates into two distinct, yet closely related design requirements:

1. The communication efficiency has to be maximized through the design of simple yet flexible and effective communication protocols and functions.
2. Those protocols and functions have to be implemented by small chips with limited computational and memory resources.

Simultaneous achievement of these objectives necessitates some kind of cross-layer protocol optimization in which the MAC layer would use the information obtained from, and control the operation of the PHY layer. At the same time, optimal operation of the upper, network and transport layers requires the knowledge of appropriate information from both the PHY and MAC layers. Again, such tight integration is not too common in ad hoc networks. An important consequence of the requirement for energy efficiency is the limited transmission range of most sensor node radio subsystems; few real devices have a transmission range of more than 100 meters (300 feet), and ranges of 10 meters (30 feet) and even less are not uncommon.

⌘ **PROTOCOL EFFICIENCY:** Regarding communication protocols, the main sources of inefficiency are packet collisions, but also overly complex handshake protocols, receiving packets destined for other nodes, and idle listening to the medium [22]. Actual power consumption of sensor nodes, often called motes,

depends mostly on the radio subsystem and its operating mode. In most (but not all) cases, transmitting and receiving use about the same amount of energy, depending on the power level used for transmission. However, most savings can be made by putting the node to sleep, when power consumption drops by one to two orders of magnitude, depending on the hardware [23, 24].

☞ **USE OF REDUNDANT SENSORS:** Since nodes are small and cheap to produce and the network lifetime needs to be maximized, it is often feasible to deploy the sensors in a given physical space in much larger numbers than necessary to obtain the desired rate of information flow. If redundant sensors are used, they can be periodically sent to sleep in order to minimize their duty cycle, which extends the lifetime of individual sensors and of the entire network and reduces or eliminates the need for operator intervention, thus reducing the operational cost of the network [25]. The use of redundant sensors has profound implications on the design of MAC protocols.

☞ **NODE SPECIALIZATION:** Another important distinction is related to the role of individual nodes. An ad hoc network allows its nodes to choose the specific role, or roles, they would like to play – i.e., data source, destination, or intermediate router – at any given time. In most cases, a node is free to switch to a different role, or roles, whenever it finds appropriate or is instructed to do so by the specific application currently executing on it. On the contrary, nodes in a sensor network have specific roles that do not change often, or never change at all. Most of the nodes act as sensing nodes, some act as intermediaries which route the traffic and (possibly) perform some administrative duties, and a small number of nodes (sometimes only a single node) act as the network sink (or sinks) toward which all the sensed data ultimately flows [26]. A group of sensor nodes under the control of an intermediary is sometimes referred to as a sub-network or cluster, while the intermediary itself is known as cluster head. The number of intermediate levels interposed between the sensing nodes and the network sink(s) depends on a number of variables such as the size of the network, the size of the physical space which the network has to monitor, the transmission range of individual nodes, and (to some extent) the actual MAC protocol used.

☞ **TRAFFIC CHARACTERISTICS:** The traffic in sensor networks is rather asymmetric, as the bulk of it flows from the sensing nodes toward the network

sink (this is often referred to as the uplink direction). The traffic in the opposite direction is generally much smaller and consists of control information and, possibly, queries issued by the network sink on behalf of the corresponding sensing application [27]. Furthermore, traffic patterns in sensor networks are rather different than in ad hoc networks. For example, temperature or humidity monitoring might require periodic or nearly periodic transmissions – in essence, synchronous traffic with low data rate – while object surveillance and other event-driven sensing applications exhibit low average traffic volume and random bursts with considerably higher peak rates. Furthermore, data packets are often much smaller in sensor networks. Original data from sensing nodes typically consists of only a few data values reported by appropriate sensors. Intermediate nodes may choose to aggregate those values in order to improve energy efficiency and reduce bandwidth and energy consumption; data aggregation is more common in networks with a larger number of hierarchical levels. At the same time, the number of sensor nodes and their spatial density may be very large, depending on the size of the space to be monitored and the requirements of the sensing application.

✎ **QUALITY OF SERVICE REQUIREMENTS:** Delay considerations are of crucial importance in certain classes of applications, for example, in military applications such as battlefield communications and detection and monitoring of troop movement, or in health care applications where patients in special care units must be monitored for important health variables (via ECG or EEG) due to a serious and urgent medical condition. Maintaining prescribed delay bounds in a network of resource-constrained nodes with limited transmission range is a complex issue. Low delays can be achieved either by bandwidth reservation, as utilized in variations of the TDMA approach, or by some kind of admission control that will prevent network congestion, if the CSMA approach is used. At the same time, the requirement for maximum throughput is relaxed due to the following. First, the exact value of the throughput requirement is usually prescribed by the sensing application, unlike general networks where the goal is to obtain as much throughput as possible. Second, energy efficiency dictates the use of protocols that incorporate power control, which will strive to keep the nodes inactive for as long as possible [25]. In order to obtain the desired throughput, it suffices to adjust the mean number of active nodes. Even packet losses can be

catered to in this manner, since we don't care whether a given packet from a given node will reach the network sink – as long as the sink receives sufficient number of packets from other nodes. Any packet loss can be compensated for (in the long term) by varying the mean number of active nodes. In a certain sense, fairness is not needed at the node and packet level as long as it is maintained at the cluster level [28]. On the contrary, fairness at the node/packet level is important in ad hoc networks.

✎ ***DIFFERENCES FROM AD HOC NETWORKS:*** The requirements outlined above lead to a number of important differences between sensor networks and ad hoc networks, most notably the following:

- ✎ Power efficiency and lifetime maximization are the foremost requirements for sensor networks.
- ✎ Self-organization is important in both ad hoc and sensor networks. In the former case, this is due to dynamicity and node mobility, which cause frequent topology changes and make self-organization more difficult; in the latter, this is mostly caused by sensor nodes exhausting their battery power (i.e., dying), although mobile sensors are used in some applications.
- ✎ Throughput maximization is often required in ad hoc networks but is not too common in sensor networks.
- ✎ Delay minimization is typically assigned much higher priority in sensor networks than in their ad hoc siblings. The use of redundant sensors allows for a certain level of fault tolerance; on the contrary, packet losses are intolerable in ad hoc networks.
- ✎ Scalability is an important issue due to the potentially large number of sensors; scalability is also important in ad hoc networks, but it is limited by the available bandwidth and the desired throughput.
- ✎ Nodes in ad hoc networks are often mobile, while most sensor networks have no mobile nodes.

In more than one sense, wireless ad hoc networks are a class of networks with flexible topology but without infrastructure, that should cater to all kinds of networking tasks. On the other hand, sensor networks are highly specialized networks that perform a rather restricted set of tasks under severe computational and communication restrictions.

Summary:

This chapter describes classification of Networks on the bases of Wired, Wireless and Hybrid Communication Network. The various constraints, characteristics and parameters of WSN are also discussed.



Chapter 3



WSN Simulators



This chapter depicts Simulation platform for Wireless Sensor Network (WSN) and Network Simulation using OPNET. The aim of Simulation platforms for WSN is to describe the characteristics of WSN simulation, briefly describe the current popular WSN simulation tools, detail the OPNET and simulate a comparative simulation study regarding nature of wired and wireless network.

3.1 Simulation Platforms for WSN

WSN is one of the most worthwhile communication network and simulation tools for WSN are increasingly being used to study sensor webs and to test new applications and protocols in this evolving research field. There is always an overriding concern when using simulation that the results may not reflect accurate behavior. However due to the associated cost, time and complexity involved in implementation of such networks, developers prefer to have first-hand information on feasibility and reflectivity crucial to the implementation of the system prior to the hardware implementation [1]. This is especially true in sensor networks, where hardware may have to be purchased in large quantities and at high cost. Even with readily available sensor nodes, testing the network in the desired environment (running real experiments on a test-bed) can be a time consuming and difficult task.

Besides, repeatability is largely compromised since many factors affect the experimental results at the same time. It is hard to isolate a single aspect. Simulation-based testing can help to indicate whether or not the time and monetary investments are worthwhile. Simulation is, therefore, the most common approach to developing and testing new protocol and applications for sensor networks. This leads to the recent boom of simulator development [2]. There are a number of advantages to this approach including lower cost, ease of implementation, and practicality of testing large-scale networks.

3.2 The Characteristics of WSN Simulation

Wireless sensor network is a highly application-oriented network type. The characteristics of the simulation in the following areas described in Table 3.1 are different from the existing wired and wireless network simulation [3].

In order to effectively develop any protocol based on simulations, it is important to know the different tools available and their benefits and drawbacks. Given the facts that simulation is not perfect and that there are a number of popular sensor network simulators available, thus making different simulators accurate and most effective for different situations/applications. It is crucial for a developer to choose a simulator that

best fits the application [4]. However, without a working knowledge of the available simulators, this can be a challenging task. Additionally, knowing the weaknesses of available simulators could help developers to identify drawbacks of their own models, when compared with these simulators, thus providing an opportunity for improvement. It is thus imperative to have a detailed description of a number of the more prominent simulators available. In this chapter, several main-stream WSNs simulators are described in more detail.

Sr. No.	Characteristics	Comments
1.	Simulation Scale	For the traditional wired network, the entire network performance can be greatly simulated by the use of the limited and represented node topology. However, due to the disadvantages of large redundancy wireless sensor network and high-density node topological structure types, the overall performance of WSN cannot be analyzed by the limited number of nodes [5]. So, in the WSN simulation scale, a large number of nodes in parallel computing must be considered [6, 7].
2.	Simulation Target	The traditional wired and wireless network simulation analysis mainly focus on quality of service (QoS), such as network throughput, end-to-end delay, and packet loss rate, which is not the main target of the analysis in most applications of wireless sensor networks [6]. The life and energy consumption analysis of the node are the most important objectives of the analysis [6, 8].
3.	Node Characteristics	The WSN nodes are influenced not only by the noise, interference, and known destruction factors, but also affect the instability of the nodes [5]. This is because of the limited capability of the node itself, coupled with the easy to fail feature (e.g. node energy exhausted), which have exacerbated the uncertainty of the network, and they are rarely seen in the previous network system [6, 7].

Table 3.1 Characteristics of WSN Simulations

3.2.1 NS-2

NS-2 (Network Simulator-2) [9, 10, 11] is a well-known network simulator for discrete event simulation. Simulations are based on a combination of C++ and OTcl [12]. NS-2 includes a large number of simulated network protocols and tools used for simulating transport control protocol (TCP), routing algorithm, multicast protocol over the wired or wireless (local connection or via satellite connection) networks [9]. NS-2 is committed to OSI model simulation, including the behaviour of physical layer and it is a free open source software and available for free download [11]. NS-2 has a number of limitations [13]:

- 1) It puts some restrictions on the customisation of packet formats, energy models, MAC protocols, and the sensing hardware models, which limits its flexibility.
- 2) The lack of an application model makes it ineffective in environments that require interaction between applications and the network protocols.
- 3) It does not run real hardware code.
- 4) It has been built by many developers and contains several inherent known and unknown bugs.
- 5) It does not scale well for WSNs due to its object-oriented design.
- 6) Using C++ code and OTcl scripts make it difficult to use.

Actually, NS-2 was not initially designed to simulate wireless sensor network, but a few research groups had extended NS-2 in order to enable it to support wireless sensor network simulation, including sensor model, battery model, a small stack, and hybrid simulation tools [14]. It is extensible, but not very scalable because of the split-programming model and object-oriented structure. In addition, because NS-2 can simulate very detailed data packet close to the exact number of running packets, it is unable to carry out large-scale network simulation [9].

To overcome the above drawbacks the improved NS-3 simulator [15] was developed. NS-3 supports simulation and emulation. It is totally written in C++, while users can use python scripts to define simulations. Hence, transferring NS-2 implementation to NS-3 require manual intervention. Besides the scalability and performance improvements, simulation nodes have the ability to support multiple radio interfaces and multiple channels. Furthermore, NS-3 supports a real-time schedule that makes it possible to interact with real systems [15]. For example, a real network device can emit and receive NS-3 generated packets.

3.2.2 GloMoSim

Global Mobile Information System Simulator (GloMoSim) [16-18] is a scalable simulation environment for large wireless and wired communication networks. GloMoSim follows the idea of the OSI reference model by using a layered approach. For the communication between the different simulation layers a standard API is used so that new models and layers can be rapidly exchanged and integrated. The node aggregation technique is introduced into GloMoSim to give significant benefits to the simulation performance. In GloMoSim, each node represents a geographical area of the simulation. Hence the network nodes which a particular entity represents are determined by the physical position of the nodes.

GloMoSim uses the parallel discrete-event simulation capability provided by Parsec (Parallel Simulation Environment for Complex Systems) [19], a c-based simulation language for sequential and parallel execution of discrete-event simulation models. Both, GloMoSim as well as Parsec were developed by the Parallel Computing Lab at UCLA. GloMoSim can be run on Windows as well as Unix derivatives.

As in NS-2, GloMoSim uses an object-oriented approach, however for scalability purposes; each object is responsible for running one layer in the protocol stack of every node. This design strategy helps to divide the overhead management of a large-scale network. Though it is a general network simulator, GloMoSim currently supports protocols designed purely for wireless networks. GloMoSim is effectively limited to IP networks because of low level design assumptions and it is not capable of simulating sensor networks accurately [20]. Therefore, it suffers the same problems as Ns-2, the packet formats, energy models, and MAC protocols are not representative of those used in wireless sensor networks.

Moreover, GloMoSim does not support phenomena occurring outside of the simulation environment, all events must be gathered from neighbouring nodes in the network. Finally, GloMoSim stopped releasing updates in 2000 and released a commercial product called QualNet.

3.2.3 QualNet

QualNet (Quality Networks) work from Scalable Network Technologies (a spin out company from UCLA) is a commercial network simulator tool [21] that is derived from GloMoSim. QualNet significantly extends the set of models and protocols supported by GloMoSim. It has a dedicated and fully implemented protocols and modules for both the wired and wireless scenarios including ad hoc, cellular and satellite

models. QualNet is a discrete-event simulator, as such, it is event driven and time aware. It uses a layered architecture that is run by each node. When a protocol resides in a particular layer at one node, the packets are passed down crossing the remaining layers at the sending node, across the network, and then up to the protocol stack at the receiving node. The basic version of QualNet software comes with the standard library which offers the most common models and protocols necessary for both wired and wireless network modelling for research and industrial purposes. Many other libraries can be purchased separately including the MANET library which provides specific components for ad hoc networks, energy and mobility models other than those already present in the standard library; and a QoS library which includes specialised protocols for implementing quality of service. The authors of QualNet claim it to be the most complete network simulator, in terms of available protocols, models and tools for mobile ad hoc networks. Another key advantage is that the authors provide the C source code for all the components, modules, models and protocols; this allows the customers to fully modify or tweak the models as well as to better understand the working of models.

QualNet has a modular design and an intuitive GUI that make it easy to use to learn and modify. The QualNet Developer software suite consist of five different components put together to form a complete solution for any type of network analysis.

3.2.4 TOSSIM

TOSSIM (TinyOS Mote Simulator) [22] is an open-source operating system specially developed for the wireless embedded sensor networks. There are few hardware platforms available for TinyOS, some commercial and some non-commercial. TinyOS release includes a simulator called TOSSIM. It is built especially for Berkeley Mica Mote platform. TOSSIM is an emulator rather than a simulator, as it runs actual application code. Simulated application code can be transferred directly to the platform, but it might not run in a mote as it runs in a simulation due to the simplifying assumptions in TOSSIM.

Figure 3.1 shows the working flow of TOSSIM. The TOSSIM architecture is consisted of 5 segments: Frames, Components, Models, Services and Events.

TOSSIM is a very simple but powerful emulator for WSN. Each node can be evaluated under perfect transmission conditions, and using this emulator can capture the hidden terminal problems. As a specific network emulator, TOSSIM can support thousands of nodes simulation. This is a very good feature, because it can more accurately simulate the real world situation. Besides network, TOSSIM can emulate radio

models and code executions. This emulator may be provided more precise simulation result at component levels because of compiling directly to native codes.

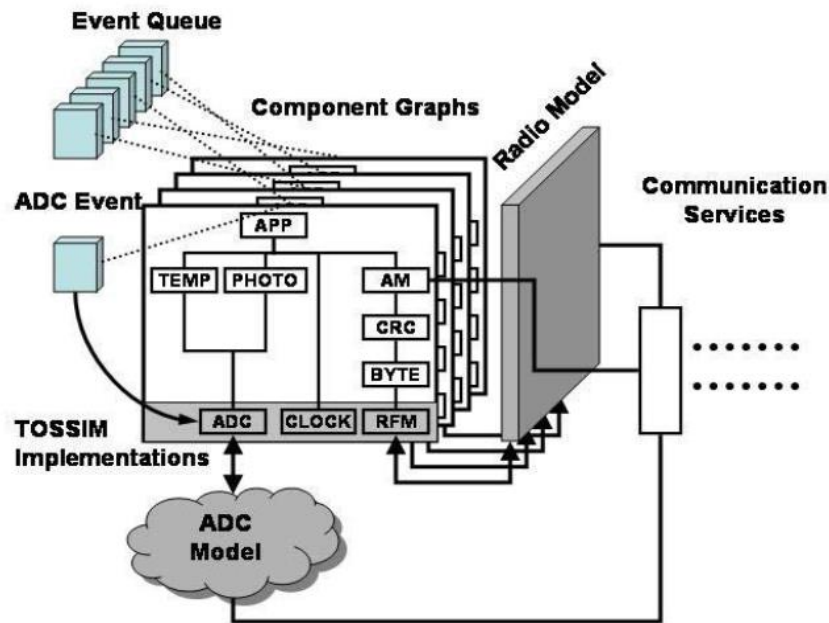


Figure 3.1: TOSSIM Architecture [23]

TOSSIM is a bit-level discrete event network emulator built in Python [24], a high-level programming language emphasizing code readability, and C++. People can run TOSSIM on Linux Operating Systems or on Cygwin on Windows. TOSSIM also provides open sources and online documents.

Developers had set four requirements for TOSSIM: scalability, completeness, fidelity and bridging. To be scalable, a simulator should manage networks of thousands of nodes in a wide variety of configurations. To achieve this, each node in TOSSIM is connected in a directed graph where each edge has a probabilistic bit error. For completeness, a simulator must capture behavior and interactions of a system at a wide variety of levels. And for fidelity, a simulator must capture behavior of a network with a subtle timing of interactions on a mote and between motes. Requirement for bridging is met as the simulated code runs directly in a real mote. [23]

The goal of TOSSIM is to study the behavior of TinyOS and its applications rather than performance metrics of some new protocol. Hence, it has some limitations, for instance, it does not capture energy consumption. Another drawback of this framework is that every node must run the same code. Therefore, TOSSIM cannot be used to evaluate some types of heterogenous applications.

3.2.5 Avrora

Avrora is a command-line framework capable of simulating and analysing programs developed for MEMSIC Mica2 and MicaZ sensor platforms. In the simulation each node has its own separated thread [25, 26].

Avrora, a research project of the UCLA Compilers Group, is an open-source cycle accurate simulator for embedded sensing programs. Unlike other simulators, that are able to simulate only specific platforms, Avrora has language and operating system independence. It provides a framework for program analysis, allowing static checking of embedded software and an infrastructure for future program analysis research. Avrora simulates a network of motes, runs the actual microcontroller programs (rather than models of the software), and runs accurate simulations of the devices and the radio communication [27].

Avrora [28-30] is an instruction-level simulator instead of just simulating software models. This approach provides an accurate simulation of devices and radio communication so that, for example, the energy usage can be predicted according to the number of clock-cycles needed for the used instructions, which removes the gap between TOSSIM and ATEMU. The codes in Avrora run instruction by instruction, which provides faster speed and better scalability. Avrora can support thousands of nodes simulation, and can save much more execution time with similar accuracy. Avrora is implemented in Java which offers great flexibility and portability because the simulation of machine code is operating system independent.

Avrora lacks an integrated graphical user interface so that everything has to be done manual on the command line.

3.2.6 OMNET++

The Objective Module Network Test-bed in C++ (OMNeT++) [31] is a component-based, modular and open-architecture simulation environment with strong GUI support and an embeddable simulation kernel. OMNeT++ is a public source component-based discrete event network simulator [32]. The simulator mainly supports standard wired and wireless IP communication networks, but some extensions for WSN exist. Like NS-2, OMNeT++ is popular, extensible and actively maintained by its user community in the Academia who has also produced extensions for WSN simulation.

OMNET++ is becoming a popular tool and its lack of models is being cut down by recent contributions. For instance, a mobility framework has recently been released for OMNET++ [33], and it can be used as a starting point for WSN modelling. OMNeT++

provides a hierarchical nested architecture. The modules are programmed in C++; the GUI of OMNeT++ is created using the Tk library. The modules are assembled into components and models by using a high-level language (NED). Modules communicate by sending messages. The simulation configuration is managed by .ini files. There are several OMNeT++ based WSNs simulation frameworks. e.g., Castalia, MiXiM, NesCT, PAWiS, SENSIM (SensorSimulator).

Additionally, several new proposals for localization and MAC protocols for WSN have been developed with OMNeT++, under the Consensus project [34], and the software is publicly available. Nevertheless, most of the available models have been developed by independent research groups and do not share a common interface, what makes difficult to combine them. As an example, not even the localization and MAC protocols developed in the Consensus project are compatible.

3.2.7 SENS

SENS [35] was developed to solve the deficiencies of the traditional network simulators, which has been used in the field of wireless sensor network. It came after NS-2 providing some necessary changes [4]. SENS uses the system structure in the module which can be reused, as long as the interfaces between modules meet the requirements. Then both modules can be reused and replaced, even the new simulation programs can be fully developed on the basis of SENS. The differences between SENS and other simulators are that SENS uses parallel simulation and serial simulation optional modes. The system default is the serial simulation. This is to consider parallel simulation in many cases caused low efficiency, thus giving users opportunity to choose according to their needs.

SENS is a customizable sensor network simulator for WSN applications [1]. Multiple component implementations in SENS offer varying degrees of realism. Users can assemble application-specific environments; such environments are modelled in SENS by their different signal propagation characteristics. The same source code that is executed on simulated sensor nodes in SENS may also be deployed on actual sensor nodes, this enables application portability.

Simulation/Programming language is written in C++. There exists a thin compatibility layer to enable direct portability between SENS and real sensor nodes.

3.2.8 MATLAB: TrueTime toolbox

TrueTime [36] is a freeware Matlab/Simulink-based simulator for networked and embedded real-time control systems. This toolbox facilitates co-simulation of controller

task execution in real-time kernels, network transmissions and continuous plant dynamics.

This toolbox provides possibility to write tasks as M-files, C++ functions or call Simulink block diagrams from within the code functions. TrueTime blocks include generally used networks as Ethernet, CAN, TDMA, FDMA, Round Robin or Switched Ethernet. It supports simulation of Wireless networks (802.11b/g WLAN and 802.15.4 ZigBee) and battery-powered devices. In a brief description, TrueTime is a small library of simulation blocks which extends usability of Matlab/Simulink to simulate discrete network process control.

Every TrueTime toolbox simulation scheme should contain three crucial parts: TrueTime kernel (computer, I/O device or some embedded system), TrueTime network (network model) and a controlled process. There is also an optional part TrueTime battery (all blocks are shown on Figure 3.2).

TrueTime kernel is responsible for I/O and network data acquisition or data processing and calculations. It can realize a control algorithm/logic and it is the “brain” of every device. It uses several simple M-files (modified by us to satisfy our needs) which handle all mentioned operations. In the kernel can be executed several independent tasks (periodic, non-periodic) which can cooperate on the same goal.

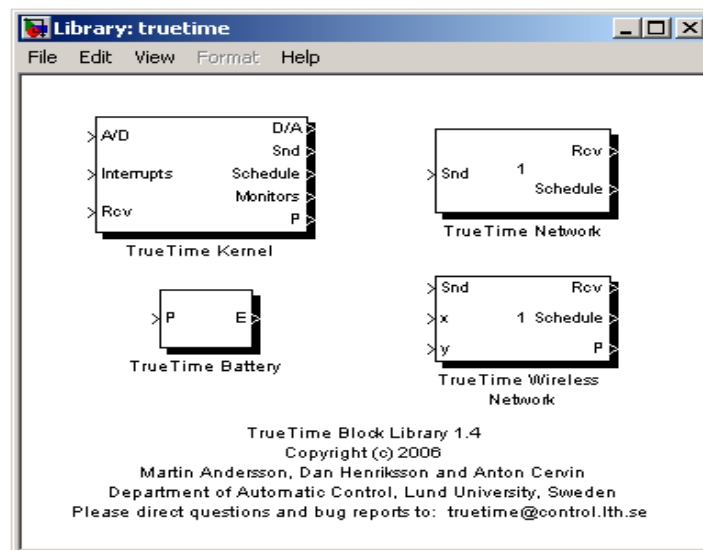


Figure 3.2: The TrueTime toolbox

One of its main features involves the possibility of co-simulation of the interaction between the real-world continuous dynamics and the computer architecture in the form of task execution and network communication. Some fundamental features in the context of this work include [37]: *i*) emulation of a wireless network and transmission

models; *ii*) representation of the kernel of a mobile node communicating through the implemented network; *iii*) representation of battery-powered devices and some basic consumption models.

In contrast to other co-simulation tools such as State flow/Simulink or Ptolemy II [38], TrueTime is not based on a mathematical modelling formalism. Rather, a TrueTime simulation is programmed in much the same way as a real embedded system. The application is written in Matlab code or in C++. The main difference from real programming is that the execution/transmission times must be specified by the developer. This approach makes TrueTime a very flexible co-simulation tool. Also, the step from simulation code to production code is not that large. The main drawback is that the simulation models are not amenable to analysis.

3.2.9 OPNET

Optimised Network Engineering Tools (OPNET) [39] is an event-driven, network simulation tool, which allows an easy implementation of the all model elements. A Graphical User Interface supports the configuration of the scenarios and the development of network models. Three hierarchical levels for configuration are differentiated: network, node and process. The network domain consists of nodes, links and subnets. A node represents a network device and groups of devices, i.e. servers, workstations etc. and WLAN nodes, IP clouds etc. Links represent point-to-point and bus-links between the nodes. The node domain covers the building of blocks, also referred to as modules, including processors, queues and transceivers as well as the specification of interfaces between the modules. The process model defines the underlying protocols, is represented by finite state machines (FSMs) and is created with states and the state transitions of the node model elements. It abstracts the behavior of the network element. The packet format generator allows building any packet consisting of a real byte oriented packets on named unsorted fields. The packets definition can follow exact protocol specifications. It is easy to deploy network elements in the project editor. All parameters can be configured easily. OPNET includes available tools for link setup and mobility profiling. Simulation results can be processed and analyzed with advanced functions. The analysis of simulated data is supported by a variety of built-in functions. The source code is based on C/C++.

To build a network model the workflow centres on the Project Editor. This is used to create network models, collect statistics directly from each network object or from the network as a whole, execute a simulation and view results. See Figure 3.3.

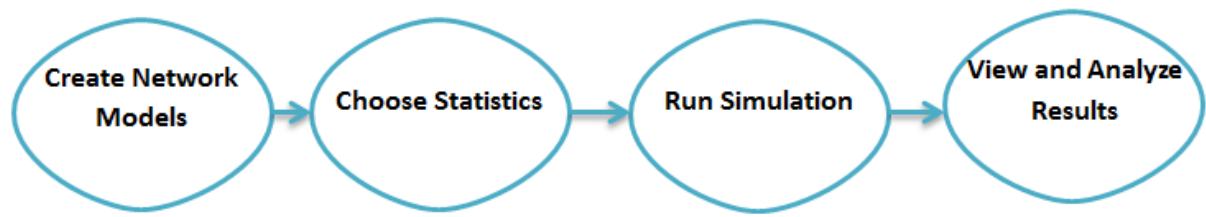


Figure 3.3: Workflow

3.2.9.1 Specification Editors

OPNET Modeler supports [39] model specification with a number of tools, called editors, which capture the characteristics of a modeled system's behavior. Because it is based on a suite of editors that address different aspects of a model, OPNET Modeler is able to offer specific capabilities to address the diverse issues encountered in networks and distributed systems. To present the model developer with an intuitive interface, these editors handle the required modeling information in a manner that is parallel to the structure of real network systems. Therefore, the model-specification editors are organized hierarchically. Models built in the Project Editor rely on elements specified in the Node Editor; in turn, when working in the Node Editor, you use models defined in the Process Editor and External System Editor. The remaining editors are used to define various data models, typically tables of values that are later referenced by process- or node-level models. This organization is reflected in the following list.

- ✘ Project Editor—Develop network models. Network models are made up of subnets and node models. This editor also includes basic simulation and analysis capabilities.
- ✘ Node Editor—Develop node models. Node models are objects in a network model. Node models are made up of modules with process models. Modules may also include parameter models.
- ✘ Process Editor—Develop process models. Process models control module behavior and may reference parameter models.
- ✘ External System Editor—Develop external system definitions. External system definitions are necessary for cosimulation.
- ✘ Link Model Editor—Create, edit, and view link models.
- ✘ Packet Format Editor—Develop packet formats models. Packet formats dictate the structure and order of information stored in a packet.

- ✘ ICI Editor—Create, edit, and view interface control information (ICI) formats. ICIs are used to communicate control information between processes.
- ✘ PDF Editor—Create, edit, and view probability density functions (PDFs). PDFs can be used to control certain events, such as the frequency of packet generation in a source module.

3.3 Network Simulation using OPNET

As networks are being upgraded from scratch all over the world, network planning is becoming most important. Computing the viability and performance of networks in real can be very expensive and painstaking task. To ease and comfort the process of estimating and predicting a network techniques are widely used and put into practice. A variety of simulation tools (discussed above) like QualNet, NS-2, Matlab and OPNET are available for the purpose of modelling and simulation but the choice of a simulator depends upon the features available and requirements of network application. Among the various network simulators, OPNET provides the industry's leading environment for network modelling and simulation. It allows to design and study communication networks, devices, protocols, and applications with flexibility and scalability. It provides object oriented modelling approach and graphical editors that mirror the structure of actual networks and network components.

The analysis in [40] helped to estimate and optimize the performance of wired and wireless networks using proposed optimization techniques. In [41], performance of wireless and wired networks as well as comparison is evaluated using OPNET simulation tool.

3.4 Simulation Setup and Results

Simulation was carried out for wired, wireless and hybrid network. For wired network, collision count, traffic received, delay, throughput is studied while for wireless network, data dropped, traffic received, media access delay, and throughput is studied. For comparison of both wired and wireless networks, the performance parameter, throughput is investigated. All these performance is carried out by varying number of users. For hybrid network, delay and throughput are investigated for both wired and wireless network part by varying traffic in terms of packet bytes.

3.4.1 Wired and Wireless Network Comparison¹

The Ethernet is a multi-access network, meaning that a set of nodes sends and receives frames over a shared link. It implements the capability of transmitting and monitoring a connected bus link at a same time. It has full duplex capability. Here Ethernet network model with star topology and data rate of 10Mbps using OPNET [39] with 25, 50,100 users is setup. Scenario for wired network, created for Ethernet using 50 nodes, is shown in Figure 3.4(a). The Wireless LAN model suite includes the features of the IEEE 802.11 operating at a data rate of 10Mbps in a star topology using OPNET Modeler 14.5 is as shown in Figure 3.4(b). The network also expanded for the 25 and 100 users.

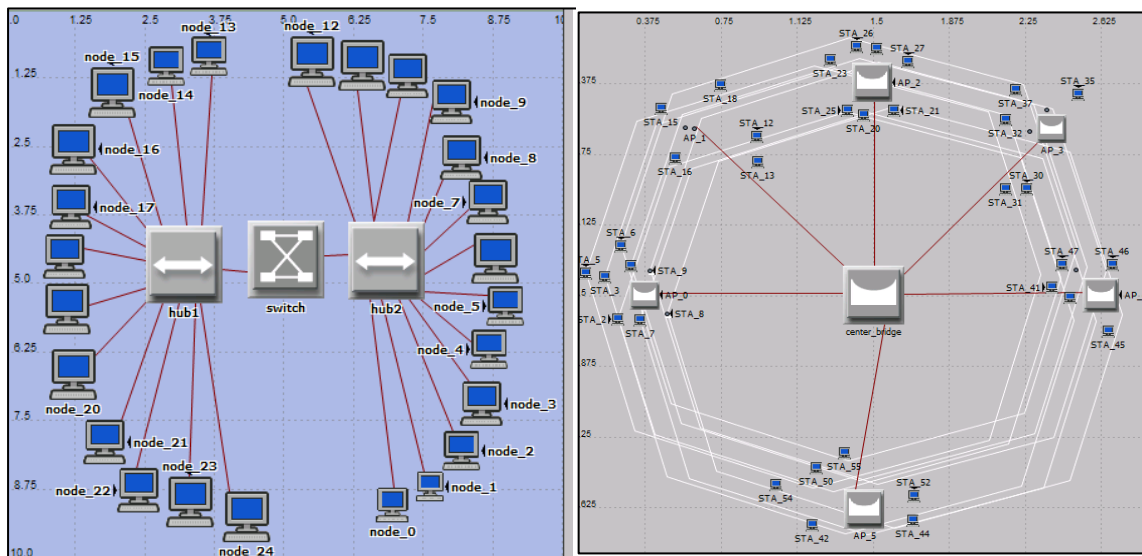


Figure 3.4(a) : Ethernet network model for 50 Ethernet stations

Figure 3.4(b) : Wireless network model for 50 users

The performance metrics evaluated are throughput, delay, data dropped, traffic received, collision count, retransmission attempts [39].

- ✂ **Throughput:** Network throughput is the average rate of successful message delivery over a communication channel. It is measured in bits per second (bit/s or bps) or in data packets per second or data packets per time slot.

¹

- ✂ **Retransmission attempts:** Total number of retransmission attempts by all WLAN MAC in the network until either packet is successfully transmitted or it is discarded as a result of reaching short or long retry limit.
- ✂ **Collision Count:** Total number of collisions encountered by this station during packet transmissions.
- ✂ **Data Dropped:** Total higher layer data traffic (in bits/sec) dropped by the all the WLAN MACs in the network as a result of consistently failing retransmissions. This statistic reports the number of the higher layer packets that are dropped because the MAC couldn't receive any ACKs for the (re)transmissions of those packets or their fragments, and the packets' short or long retry counts reached the MAC's short retry limit or long retry limit, respectively .

From the Figure 3.5 it is observed that the received bit rate for wired network is approximately equal to the sent bit rate for small number of users. As the number of user increases, more traffic was sent and received. As the number of users increases the hub switch becomes overloaded and cannot deliver all the traffic that it received.

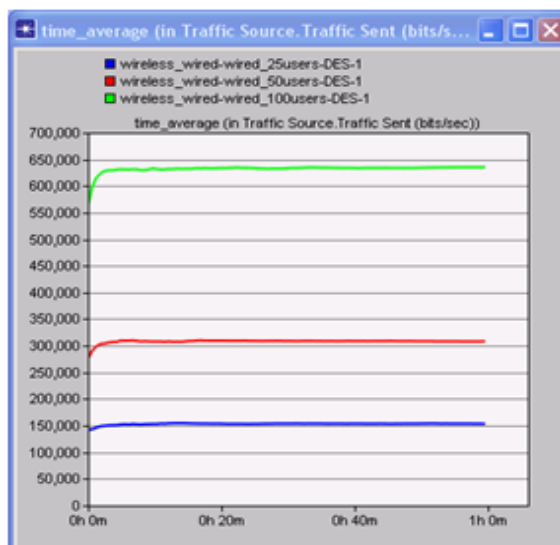


Figure 3.5(a) : Traffic sent (bits/sec) of different scenarios having 25,50,100 users

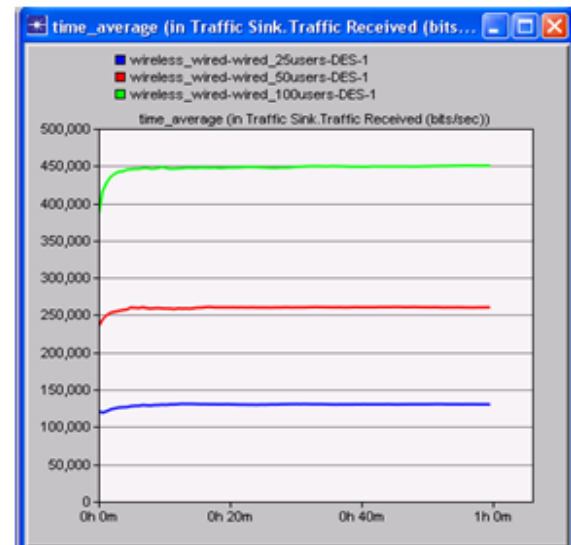


Figure 3.5(b) : Traffic received (bits/sec) of different scenarios having 25, 50, 100 users

Discrepancies between send and receive rates can be accounted for by inspecting the collision count statistic as shown in Figure 3.6. In Figure 3.7, it is observed that maximum throughput is achieved in the case when less number of users is deployed. As the number of users increases, more number of the higher layer packets that are dropped because the MAC couldn't receive any ACK for the (re) transmissions of those packets or

their fragments, and the packets' short retry limit or long retry limit, respectively. With less number of users, the overall performance of the system increases as data transmission will be faster.

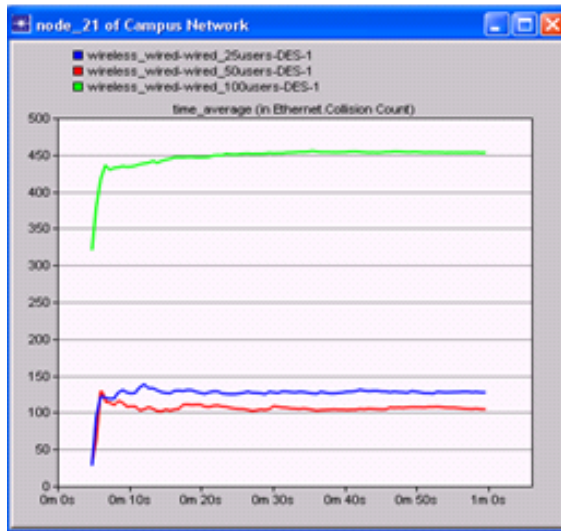


Figure 3.6 : Collision count in Wired Network

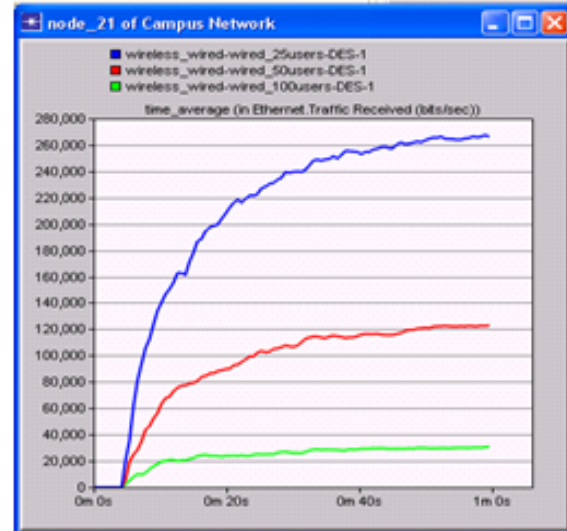


Figure 3.7 : Throughput (bits/sec) of Different scenarios on node 21 for Ethernet

As the number of users increases in WLAN, data dropped in wireless LAN increases. Some of the packets that were sent collided and require retransmissions. So as the users increase, retransmission rate increases for the more number of users which is shown in Figure 3.8(a) and Figure 3.8(b) respectively.

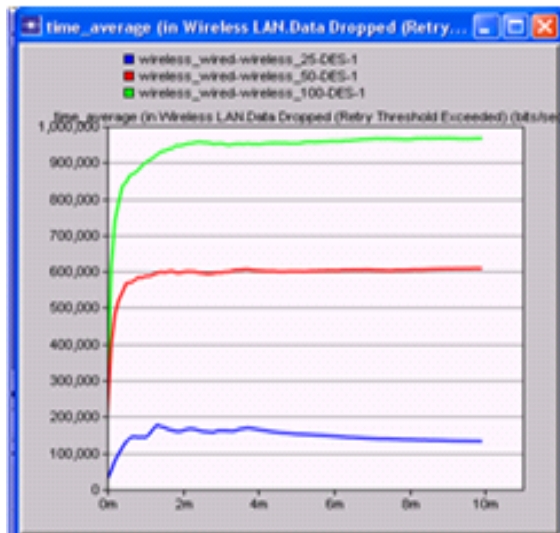


Figure 3.8 (a) : Data dropped for different scenarios for W LAN

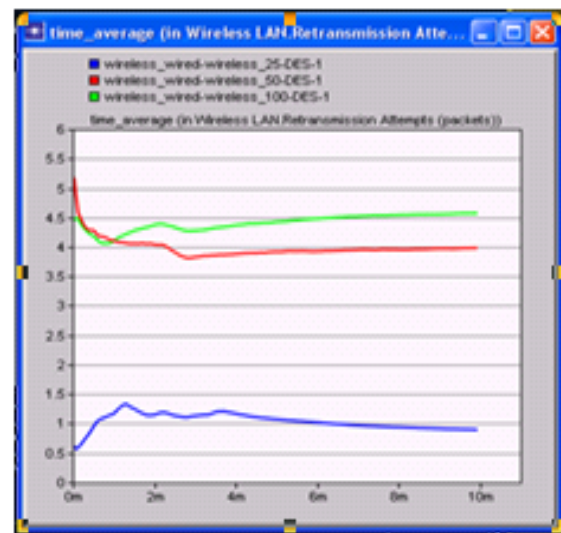


Figure 3.8(b) : Retransmission attempts of different scenarios for W LAN

As the number of user increases, collision between the user data increases and the retransmissions of the user increases which cause for the degradation in the performance

of wireless network. It can be observed from Figure 3.9 that as the number of users increases from 25 to 100 the performance of WLAN decreases.

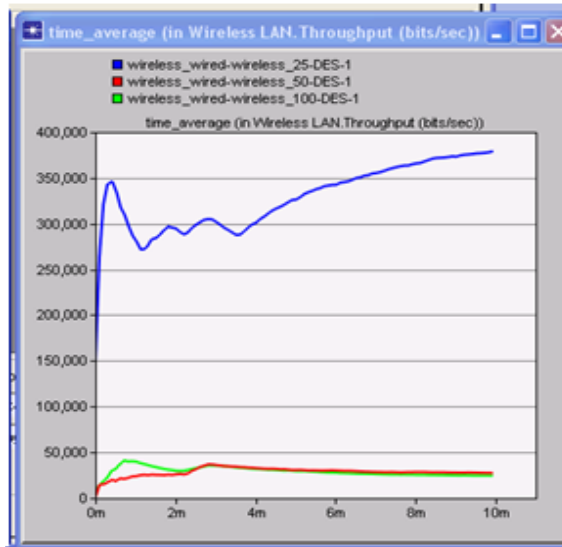


Figure 3.9 : Throughput (bits/sec) of different scenarios for WLAN

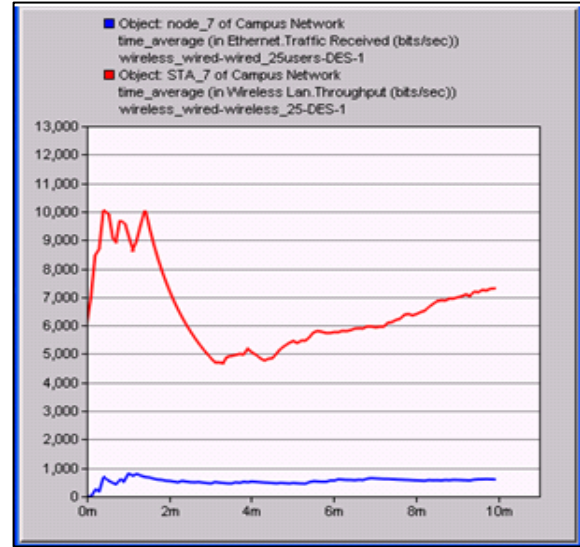


Figure 3.10(a) : Throughput of Ethernet and WLAN on node 07

Throughput comparison of wired and wireless network is performed for the number of users of 25, 50, and 100 with the data rate of 10 MBPS for both the network. The performance of the networks measured using performance of the random node present in the network. While comparing throughput of Ethernet (wired) in blue color and WLAN (wireless) in red color in Figure 3.10(a), (b), (c) for 25 users on node_07, for node 37 in 50 users scenario and node 97 for the scenario with 100 users respectively, with less number of users, the overall performance of the system increases as data transmission will be faster.

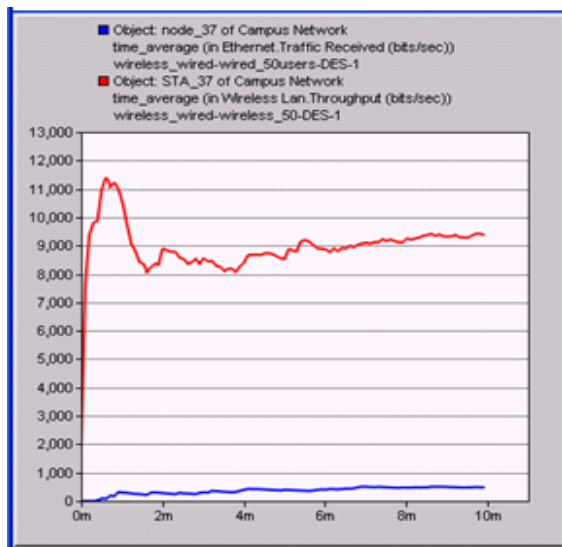


Figure 3.10(b) : Throughput of Ethernet and WLAN on node 37

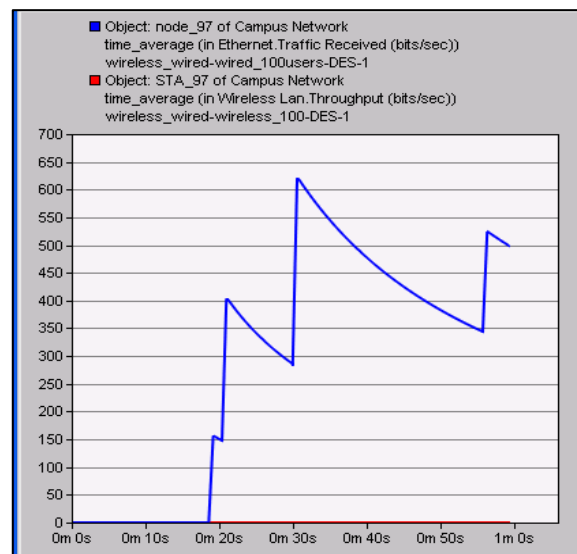


Figure 3.10(c) : Throughput of Ethernet and WLAN on node 97

Also it is observed that the throughput of Wireless LAN is greater than throughput of Ethernet for less number of users. As the number of user increases, throughput of WLAN becomes poor because of slow transmission speed and data dropped. When the number of users is increased, the throughput decreases for wireless systems, these effects associated with wireless transmission limit the SNR (Signal to Noise Ratio) and bandwidth of the received signal, and therefore the maximum number of bits that can be sent.

3.4.2 Hybrid Network

As discussed in chapter 2, the interoperability in heterogeneous networks with hybrid structure is in doubtfully a major requirement, when implementing communication scenarios for home and industrial applications. As IEEE 802.3 and IEEE 802.11 are becoming the most common and widely used LAN/WLAN standards, an interoperable architecture is required in order for communication to be treated transparently at the higher-levels. However, it has to be pointed out that from the user point of view, the entire system is seen as a black box and is expected to function equally well, independently from network heterogeneity.

It is already discussed that Hybrid networks implementation uses both ad hoc connectivity and access points. Access points receive or transmit data between both wired and wireless networks. In short, wireless networks don't replace wired networks because some wiring is still required to deploy even a simple model so an implementation of a simple hybrid network is done to evaluate the performance of global network performance and QOS tenability.

3.4.2.1 Model Outline

The proposed scenario consists of a wireless and a wired network (hybrid network). Figure 3.11 shows the structure of the model or the deployment. The purpose of the scenario is to demonstrate the inter-communication between the wireless and wired network.

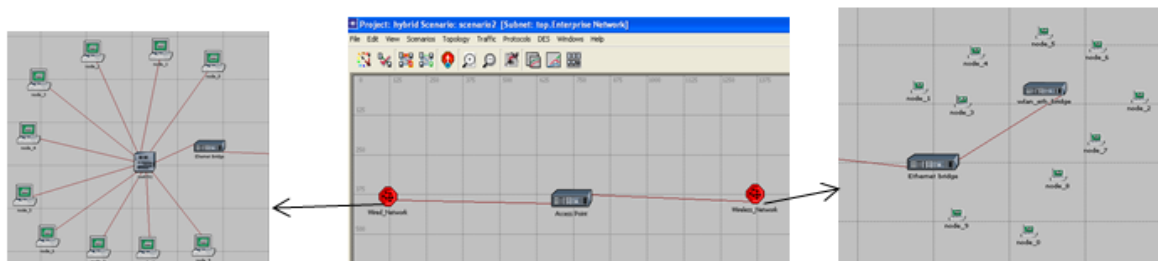


Figure 3.11: Hybrid Network

Two data traffic patterns were simulated; heavy traffic and light traffic (by adjusting packet size in bytes) as shown in Table 3.2.

Traffic Type	Packet Size (in bytes)
Light Traffic	1000
Heavy Traffic	10000

Table 3.2 Traffic Specification

As shown in Figure 3.12(a) and (b), it is obvious that the throughput for light traffic in wired and wireless network performance same but for heavy traffic, throughput in wired network is somewhat greater than wireless network part by considering segmentation.

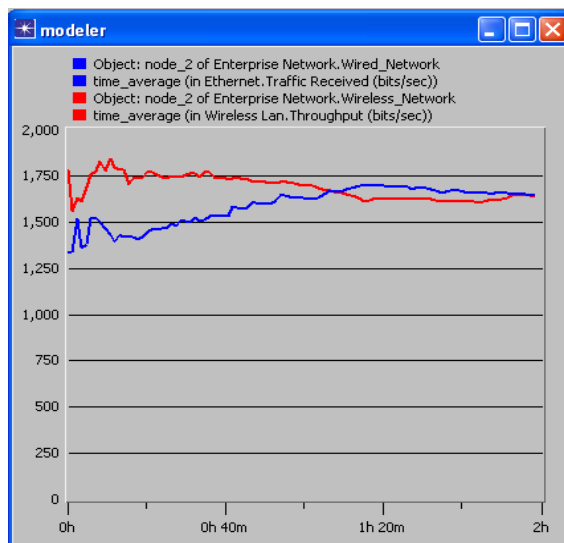


Figure 3.12(a) : Light Traffic

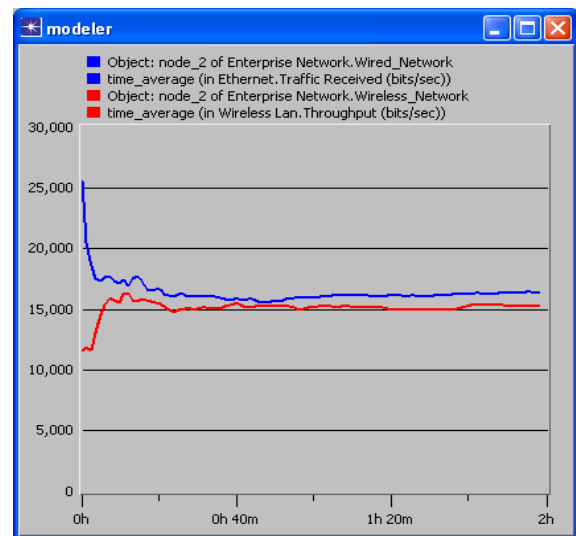


Figure 3.12(b) : Heavy Traffic

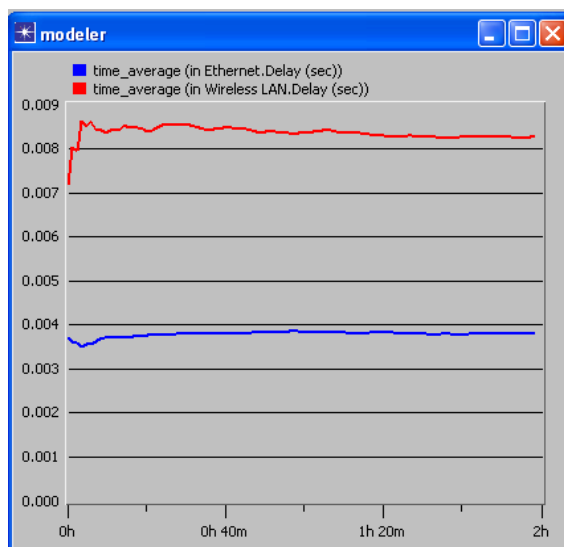


Figure 3.13 (a) : Delay for Light Traffic

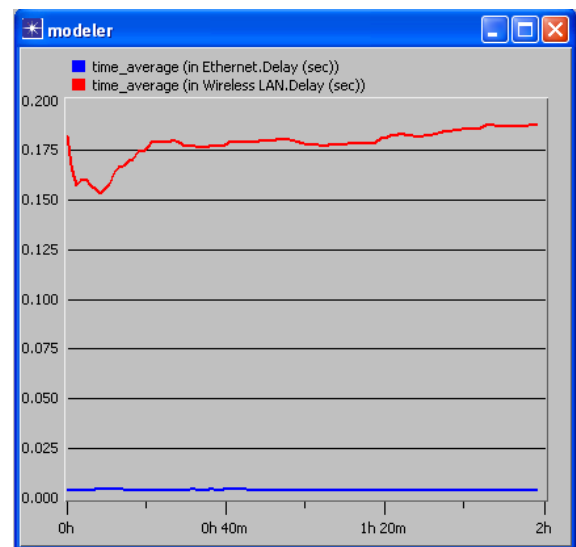


Figure 3.13(b) : Delay for Heavy Traffic

Same thing can be observed in below Figure 3.13(a) and (b), analysing delay for wired and wireless network part. For light as well as heavy traffic there is less delay in wired network.

Summary:

In this chapter, different network simulators like NETWORK SIMULATOR NS-2, OPNET, OMNET++, GLOMOSIM, and QUALNET are surveyed. OPNET simulation tool was used to evaluate the performance of the Wireless and Wired Network in terms of different number of users, traffics. For high traffic and users wired network outperforms than Wireless network due to the transmission limit, SNR (signal to noise) and bandwidth of the received signal. So to improve the overall performance of the system it is better to use hybrid network which is the combination of both wired and wireless network.



Chapter 4



Development support Tools



This chapter describes an overview of Motes of the wireless sensor network (wsn). Behaviour of the Motes analysed using MATLAB True time Toolbox based on power utilization of Motes.

For the past decade, there has been rapid development and advancement in the communication and sensor technologies that results in the growth of a new, attractive and challenging research area – the wireless sensor network (WSN). A WSN, which typically consists of a large number of wireless sensor nodes formed in a network fashion, is deployed in environmental fields to serve various sensing and actuating applications. With the integration of sensing devices on the sensor nodes, the nodes have the abilities to perceive many types of physical parameters such as, light, humidity, vibration, etc. about the ambient conditions. In addition, the capability of wireless communication, small size and low power consumption enable sensor nodes to be deployed in different types of environment including terrestrial, underground and underwater. These properties facilitate the sensor nodes to operate in both stationary and mobile networks deployed for numerous applications, which include environmental remote sensing, medical healthcare monitoring, military surveillance, etc. For each of these application areas, the design and operation of the WSNs are different from conventional networks such as the internet. The network design must take into account of the specific applications. The nature of deployed environment must be considered. The limited of sensor nodes' resources such as memory, computational ability, communication bandwidth and energy source are the challenges in network design (discussed in chapter 2).

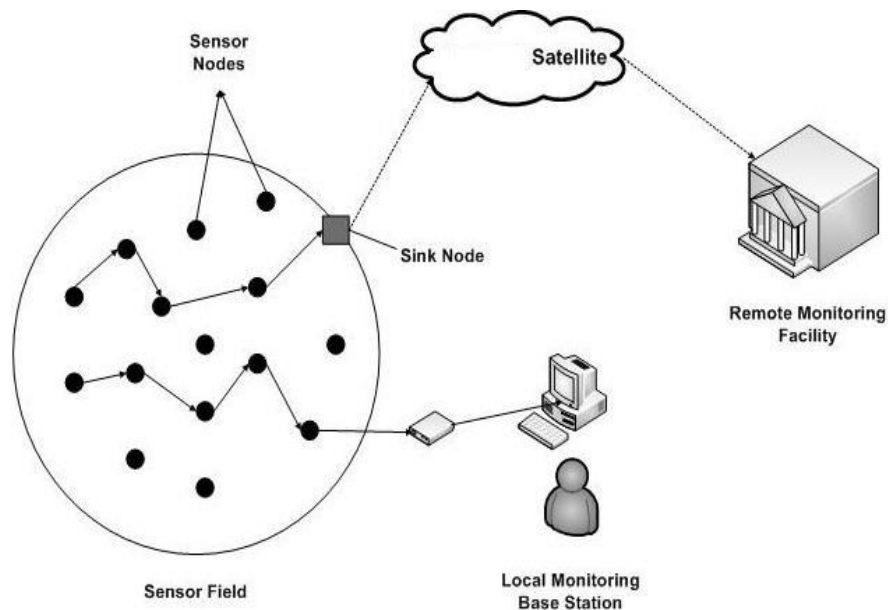


Figure 4.1: WSN [2]

WSN initially consists of small or large nodes called as sensor nodes [1]. These nodes are varying in size and totally depend on the size because different sizes of sensor nodes work efficiently in different fields. Wireless sensor networking have such sensor nodes which are specially designed in such a typical way that they have a microcontroller which controls the monitoring, a radio transceiver for generating radio waves, different type of wireless communicating devices and also equipped with an energy source such as battery. The entire network worked simultaneously by using different dimensions of sensors and worked on the phenomenon of multi routing algorithm which is also termed as wireless ad hoc networking. Each node has one or more sensors integrated on it. In addition to these sensors, a node is also equipped with a transmitter and a receiver. These transmitter and receiver are used for wireless communications with other nodes or directly with the gateway. The gateway is responsible for transmitting sensor data from the sensor patch to the remote base station that provides Wireless Ad-Hoc Network (WANET) connectivity and data logging through a local transit network. Finally, the data is available to researchers through a user interface. The scenario of Wireless Sensor Network is shown in Figure 4.1.

4.1 MOTES

Wireless Sensor Networks, also known as Sensory Networks or Sensitive Networks and abbreviated in English to WSN, consist of a series of small electronic devices that access the outdoor world by means of sensors. The name given to this type of device is “MOTE” [3], from the phrase “mote of dust”, with the idea of conveying two main concepts in just one word: their small size and the idea that they can be placed anywhere. This concept gave rise to the formation of Dust Networks and to the nickname Smart Dust. Other ways of denoting a mote include Sensory Node or Sensor Node.

Sensor networks-based monitoring applications range from simple data gathering, to complex Internet-based information systems. Motes run the network embedded programs that mainly sleep, and occasionally acquire, communicate, store and process data. The other parts of a sensor node are the microcontroller and the battery (as the energy source). Battery-powered embedded systems, such as WSN motes, require low energy usage to extend system lifetime. WSN motes must power sensors, a processor, and a radio for wireless communication over long periods of time, and are therefore particularly sensitive to energy use. Recent techniques for reducing WSN energy consumption, such as aggregation, require additional computation to reduce the cost of sending data by minimizing radio data transmissions. Larger demands on the processor

will require more computational energy, but traditional energy reduction approaches, such as multi-core scaling with reduced frequency and voltage may prove heavy handed and ineffective for motes. Battery-powered embedded systems carefully manage energy consumption to maximize system lifetime. WSNs, made up of many “mote” devices, and are often designed to operate for months without intervention.

The most important consideration for a WSN is power consumption. While the concept of WSN looks practical and exciting on paper, if batteries are going to have to be changed constantly, widespread adoption will not occur. Therefore, when the sensor node is designed power consumption must be minimized.

4.2 MATLAB

MATLAB is a high performance language for technical computing. It integrates computation, visualization and programming and easy to use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB is an interactive system while basic data element is an array that does not require dimensioning. This allows us to solve many technical computing problems, especially those with matrix vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C. The name MATLAB is an abbreviation of matrix laboratory [4].

MATLAB features a family of add-on application specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulations and many others. The MATLAB system when started, the MATLAB desktop appears, containing tools for managing files, variables and applications associated with MATLAB [5].

4.3 Simulink

Simulink is a software package for modeling, simulating and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two.

Systems can also be multirate i.e. have different parts that are sampled or updated at different rates [6]. For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click and drag mouse operations. It includes a comprehensive block library of sinks, sources, linear and nonlinear components and

connectors. We can also customize and create own blocks. For information on creating own blocks, like separate writing S-Functions. Models are hierarchical, so you can build models using both top down and bottom up approaches. We can view the system at a high level and then double click blocks to go down thorough the levels to see increasing level of model detail. This approach provides insight into how models organized and how its parts interact.

The model can be simulated using a choice of integration methods, either from the Simulink menus or by entering commands in the MATLAB Command window. The menus are particularly convenient of interactive work, while the command line approach is very useful for running a batch of simulations. For example using scopes and other display blocks, we can see the simulation results while simulation is running; the simulation results can be put in the MATLAB workspace for post processing and visualization [4].

4.4 Truetime Toolbox

Truetime is the Matlab/Simulink-based [7] simulator, which facilitates co-simulation of controller task execution in real-time kernels, network transmissions, and continuous plant dynamics. TrueTime [8] is a simulator for networked and embedded control systems that has been developed at Lund University since 1999.

TrueTime supports many network types (Wired: Ethernet, CAN, TDMA, FDMA, Round Robin, and switched Ethernet, and wireless networks: 802.11b WLAN and IEEE 802.15.4) and it is widely used to simulate wireless NCSs [8]. Besides the dynamic system simulation offered by Simulink, network node simulation includes simulation of real-time kernels. The user can write Matlab m-file functions that are scheduled and executed on a simulated CPU.

It is written in C++ MEX, and is an event-based simulation. External interrupts are implemented, and it is possible to write tasks as M-files or C++ functions. Another feature is the possibility to make the devices battery-powered and simulate the power drain. When installed, a separate block library, in addition to the standard Simulink library, becomes available. The TrueTime block library consists of four different blocks [7, 8] (referred in Figure 3.2).

The TrueTime Kernel block simulates the control mechanisms for the nodes in the network, i.e. it acts as the micro controller in the node. The TrueTime Network block simulates medium access and packet transmissions in a local area network. Six simple

models of networks are supported: CSMA/CD, CSMA/AMP, Round Robin, FDMA, TDMA and Switched Ethernet. The TrueTime Wireless Network block is similar to and works in the same way as the wired one. The TrueTime Battery block simulates a battery being drained and has only one parameter, the initial power. It uses simple integrator model, so it can be both charged and recharged. Because TrueTime is based on Simulink in Matlab, knowledge of Matlab programming and Simulink simulation is beneficial. The creation of models can be done by extensive use of graphical Simulink models or by writing Matlab code for different parts of the system. The TrueTime tool comes with a user manual that describes the functions of the models and how to use the built in library. TrueTime was selected as a simulation tool because of its simplicity and, admittedly, due to the familiarity of Matlab and limited knowledge of C++ by the authors. TrueTime is considered experimental software and can be downloaded for free from the TrueTime website [9].

The manual [10] describes the fundamental steps in the creation of a TRUETIME simulation. This include how to write the code that is executed during simulation, how to configure the kernel and network blocks, and what compilation that must be performed to get an executable simulation. The code functions for the tasks and the initialization commands may be written either as C++ functions or as Matlab M-files and both cases are described. [11] has discussed networked control in industrial applications and the possibility to simulate the control systems by TrueTime, the Networked Control Systems, their characteristics and possibilities, the wired and also the wireless systems.[12] uses TrueTime toolbox as a simple and easy way how to realize several network types. Also demonstrates how to setup simple TrueTime network control system with an explanation of its basic settings and parameters. In the last briefly compare simulation results of motor control system which uses different network types. [13] describes, a higher level simulation platform for WSN which is proposed based on the TrueTime toolbox. Relevant features include graphical representation of communication components, wireless communication and battery-driven operation. Special attention was paid to the 3D graphical interface, simulator interactivity and its extendibility. A TrueTime simulation model of the tunnel scenario is developed [14, 15]. The TrueTime simulator allows concurrent simulation of the physical robots and their environment, the software in the nodes, the radio communication, the network routing, and the ultra-sound navigation system.

4.5 Software Requirements

For the Matlab version, pre-compiled files are provided in the archive that is downloaded from the TRUETIME web site. The following compilers are currently supported (it may, of course, also work using other compilers):

- ✂ Visual Studio C++ 7.0 (for all supported Matlab versions) for Windows
- ✂ gcc, g++ - GNU project C and C++ Compiler for LINUX and UNIX

Before starting Matlab, set the environment variable TTKERNEL to point to the directory with the TRUETIME kernel files, \$DIR/kernel. This is typically done in the following manner:

✂ Unix/Linux: export TTKERNEL=\$DIR/kernel

✂ Windows: use Control Panel / System / Advanced / Environment Variables

Then add the following lines to your Matlab startup script. This will set up all necessary paths to the TRUETIME kernel files.

```
% addpath([getenv('TTKERNEL')])  
% init_truetime;
```

Starting Matlab and issuing the command

```
>> truetime
```

From the Matlab prompt will now open the TRUETIME block library, as in Figure 3.2. (chapter3).

4.6 Compilation

Since the TRUETIME archive contains pre-compiled files, no compilation is required to run TRUETIME with the M-file API. However, TRUETIME also supports simulations written in C++ code, which then must be compiled. In this case, you first need to configure your C++ compiler in Matlab. This can be done by issuing the command

```
>> mex -setup
```

In the setup, make sure that you change from the Matlab default compiler to a proper C++ compiler. For more detailed instructions on how to compile individual simulations refer manual. [9, 10].

4.7 Simulation Setup ¹

In the simulation model the movements of dynamically moving motes using the built in functionality of Matlab is used. The example consists of three motes, with dynamics in the x and y directions modeled using simple integrators. The motes are sent out on a mission which consists of visiting a number of checkpoints seen as red marks in the animation window in Figure 4.3.

In the window, the transmission range of the motes can be seen as large partly transparent colored circles around the smaller colored motes. The checkpoints should be visited at least once by some node in the group. When the motes are able to communicate, they tell each other where they are heading and share information on which nodes that have been visited by the group. Some of this information is visible as printouts during the execution. When a checkpoint has been visited it changes color from green to red. The motes operate according to the following algorithm [7]:

1. Read new network messages containing information of visited nodes target of the sending node
2. If (someone has the same target as we do && we have the lowest priority), then change target
3. If (heading to a place that has already been visited), then change target
4. If (arrived at target) then paint the target green && change target
5. Send new network messages to other nodes

The simulation results were carried out using TrueTime toolbox of MATLAB [5, 6]. Motes with random topology and 9 nodes were simulated. The performance was evaluated for Signal Reach, Visiting Priority using variation in transmission power of each node [16].

4.7.1 Process to run simulation model

To run the simulation following commands execution require at the command prompt of Matlab after setting the current path in Matlab for kernel

```
>>addpath (getenv ('TTKERNEL'))
>> init_truetime
>> truetime
```

¹ Published a paper: Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma “A Simulation Study of Behaviour of Wireless Motes With Reference To Parametric Variation” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 1, Issue 2,pp:91-95 ISSN 2278 – 8875, August 2012.

It will open the Truetime library, then open the model motes.mdl and init.m file. First execute init.m file then run the model motes.mdl file to visualize the topology animation and the results on the command prompt.

This simulation was carried out with following parameters shown in Table 4.1:

Parameters	Values
Transmit Power	-20 in dbm
Receiver Threshold	-48 in dbm
Path loss	3.5
Error Coding Threshold	0.03
Number Of Motes	3

Table 4.1 Simulation parameters

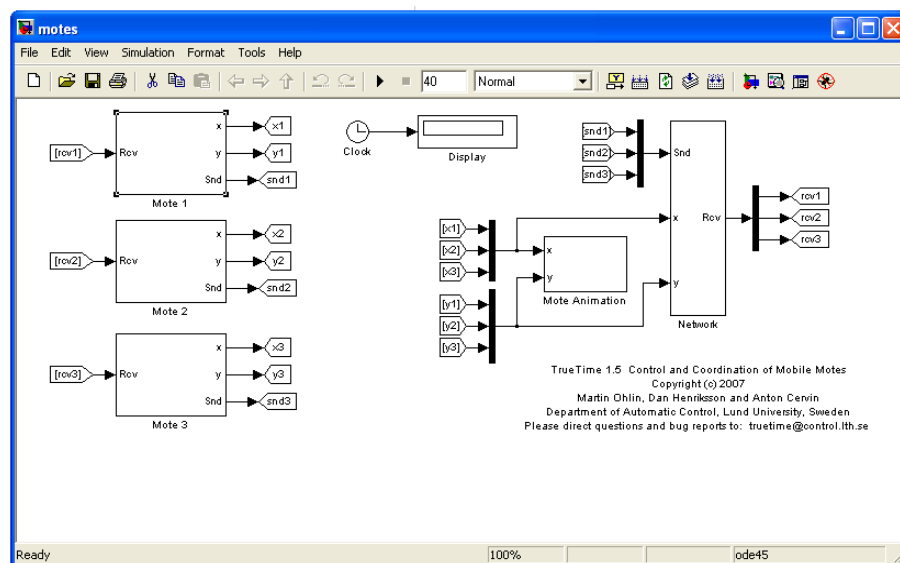


Figure 4.2: Simulink model in truetime

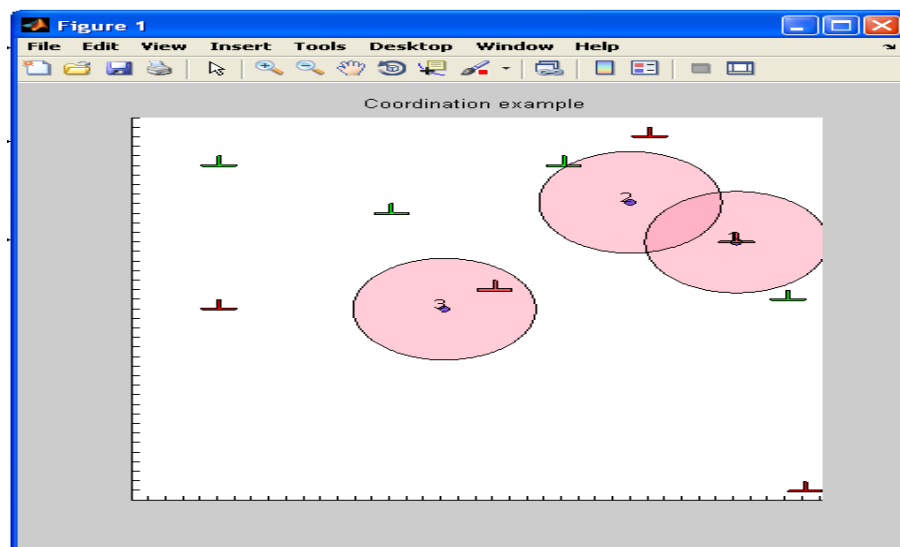


Figure 4.3: Random Network Topology

Figure 4.2 shows the Simulink model of Motes available in MATLAB True time and the random topology for wireless sensor network used for simulation is shown in Figure 4.3.

4.7.2 Simulation Results

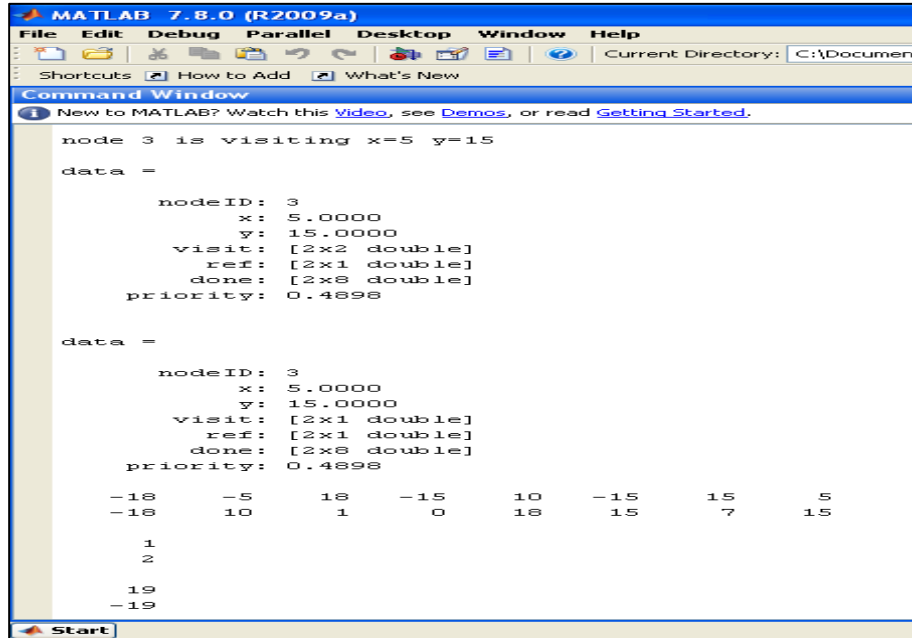


Figure 4.4: Results on Command window for MOTES Simulink model

Figure 4.4 depicts the result of simulation model in matlab command prompt which gives the information about which node visits the visiting point, with some specific priority. Also it shows the list of visited points, remaining visiting points and referred visiting point by the nodes.

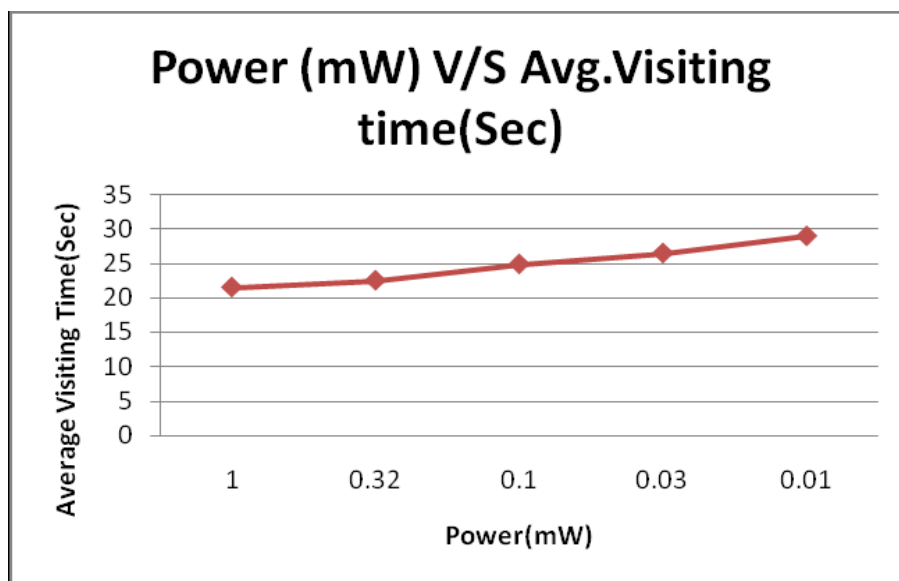


Figure 4.5: power (mW) V/S average visiting time(Sec)

Figure 4.5 shows the average visiting time taken by all the nodes to visit the visiting points in network. From this it can be observed that as the power decreases the average visiting time taken by the nodes increases.

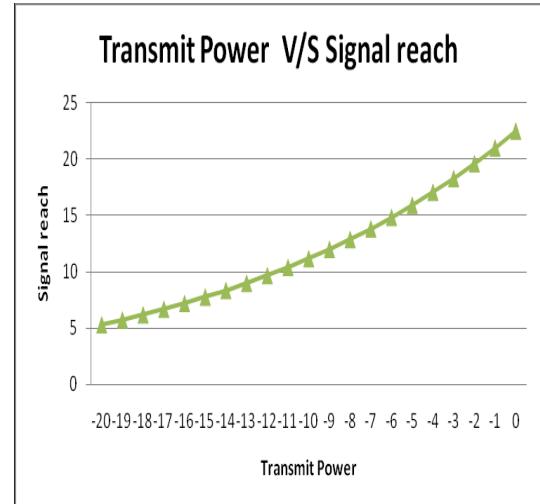
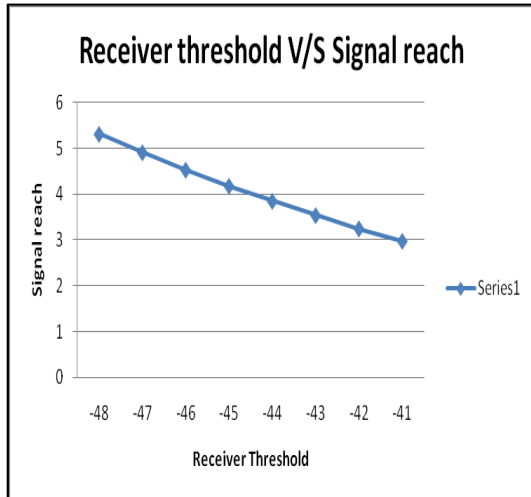


Figure 4.6: Receiver threshold level (dbm) V/S Signal Reach

Figure 4.7: Power (dbm) V/S Signal Reach

Effect of changes in Received threshold level and transmission power on Signal Reach explained in Figure 4.6 and 4.7 respectively. As the receiver threshold level increases signal reach of the node decreases. It can be observed from figure 4.7 that as transmission power increases signal reach of the nodes also increases.

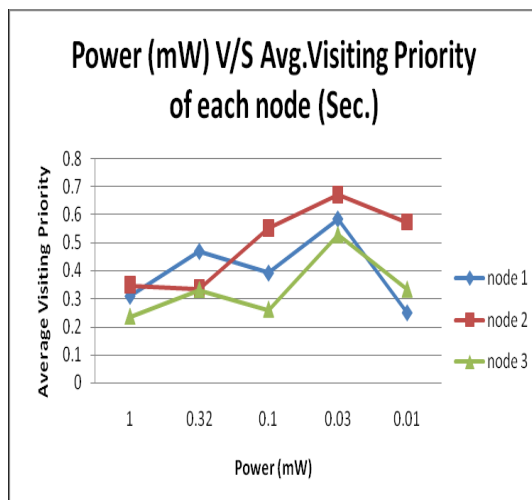


Figure 4.8: Power (mW) V/S Average Visiting Priority for each node

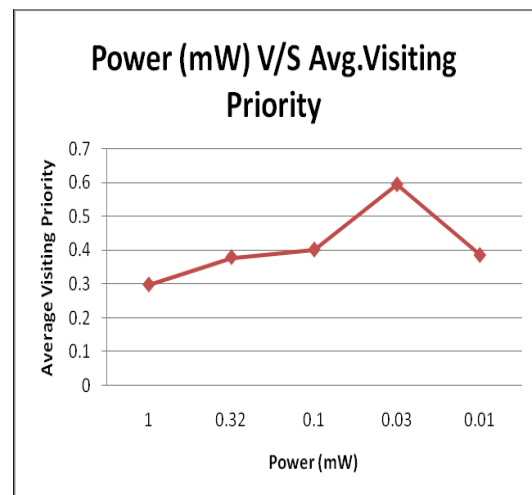


Figure 4.9: Power (mW) V/S Average Visiting Priority

Figure 4.8 depicts the change in average visiting priority of the node 1, 2, 3 with respect to change in transmission power of each node. It can be observed that as power decreases, priority also decreases for each node. Figure 4.9 shows the effect of change in power to the average visiting priority of the all nodes. As power decreases, the average priority for the nodes in network also reduces.

Summary

The power awareness issue is the primary concern within the domain of Wireless Sensor Networks (WSNs). Most power dissipation occurs during communication. From this simulation study it can be concluded that as the signal transmission power increases for each node, performance of WSN improves with signal reach. WSN consist of battery operated nodes and it is not suitable to increase the power to improve the performance of network. Suitable algorithm can be used to determine the optimized value of the transmission power for each node so that in less amount of power, performance can be improved. Also soft computing techniques can be applied to determine the optimized value of the transmission power.



Chapter 5



IEEE 802.15.4 Structure



This chapter provides an overview of the most relevant aspects of the WSN IEEE 802.15.4 protocols. WSN includes various types of the topological structure with the different parameters. In this chapter, OPNET simulator is used to evaluate the performance of two topological structures of WSN.

The Institute of Electrical and Electronics Engineers (IEEE) finalized the IEEE 802.15.4 standard in October 2003 ([1] - [4]). The standard covers the physical layer and the Medium Access Control (MAC) sub-layer of a low-rate Wireless Personal Area Network (WPAN). Even though this standard was not specifically developed for wireless sensor networks, it is intended to be suitable for them since sensor networks can be built up from LRWPANs. Sometimes, people confuse IEEE 802.15.4 with ZigBee [5], an emerging standard from the ZigBee alliance. ZigBee uses the services offered by IEEE 802.15.4 and adds network construction (star networks, peer-to-peer/ mesh networks, and cluster-tree networks), security, application services, and more. In fact, the IEEE 802.15.4 protocol targets low data rate, low power consumption, low cost wireless networking, with typically fits the requirements of sensor networks. The ZigBee/IEEE 802.15.4 protocol architecture is presented in Figure 5.1.

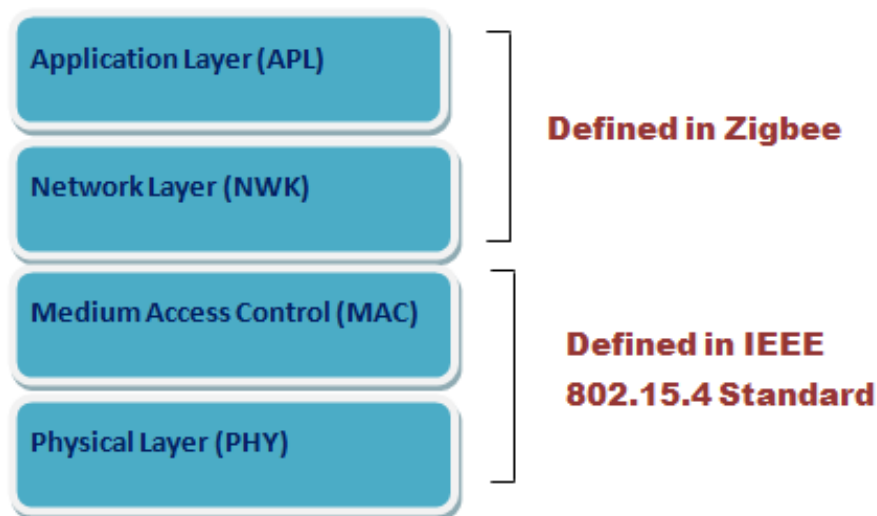


Figure 5.1: IEEE820.15.4/ZigBee protocol stack architecture

5.1 Network Devices

According to the IEEE 802.15.4 standard [6], a LR-WPAN distinguishes (on the MAC layer) two different types of devices: i) Full Function Device (FFD) and ii) Reduced Function Device (RFD).

- i. **FFD:** It supports three different roles, attending as:

- ⌘ **The Personal Area Network (PAN) Coordinator:** the principal controller of the PAN. This device identifies its own network as well as its configurations, to which other devices may be associated. This device is referred to as the Coordinator (C).
 - ⌘ **The Coordinator:** provides synchronization services through the transmission of beacons. This device should be associated to a PAN Coordinator and does not create its own network. This device is referred to as the Router (R).
 - ⌘ **The End Device:** a device which does not implement the coordinator functionalities and should associate with a C or R before interacting with the network. This device is referred to as the End Device (ED).
- ii. **RFD:** It can operate only as a device, operating with minimal implementation of the IEEE 802.15.4 protocol.

5.2 Network Topologies

The star and peer-to-peer topologies as shown in Figure 5.2 are two basic network topologies defined in the IEEE 802.15.4 standard [7].

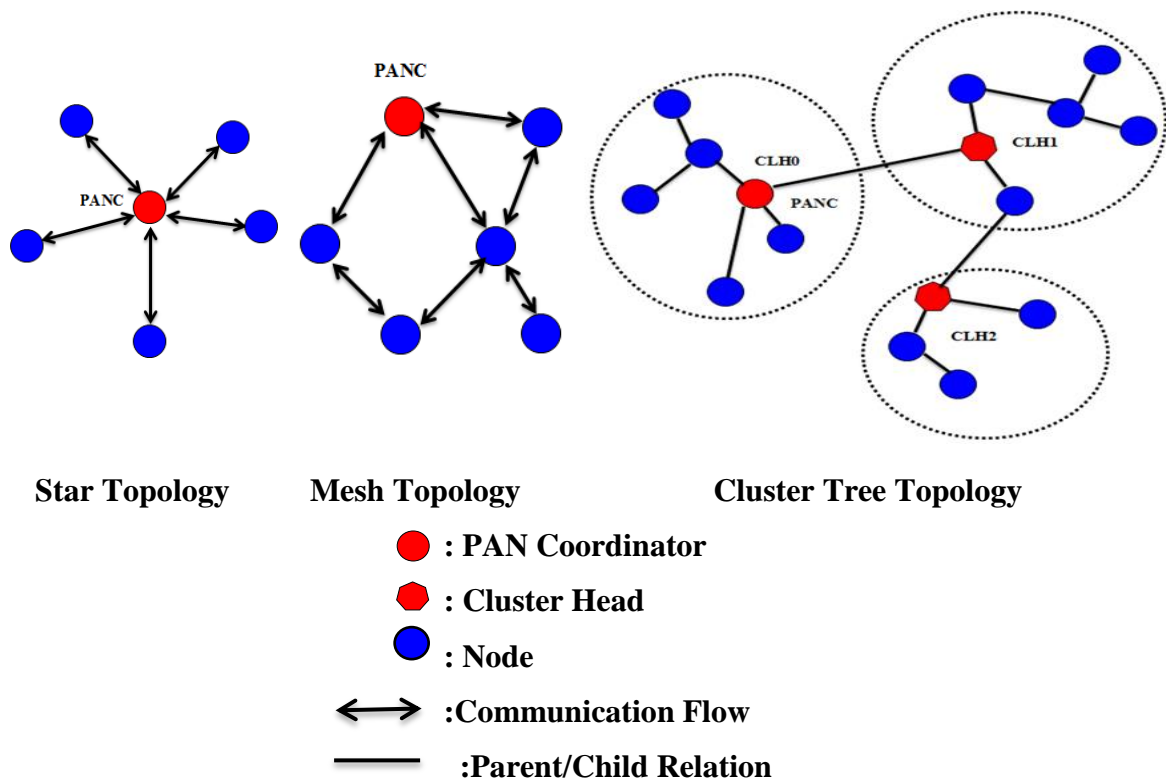


Figure 5.2: IEEE 802.15.4 Network Topologies

In the star topology, the communication is centralized and established between a PAN Coordinator and its associated devices. The main advantage of this topology is its simplicity. The peer-to-peer topology has also a PAN Coordinator; however, it differs from the star topology in that any device can communicate with any other device within its radio range. The cluster-tree topology is a special case of a peer-to-peer topology with a distributed synchronization mechanism. IEEE 802.15.4 standard in the beacon-enabled mode supports only the star topology.

5.3 IEEE 802.15.4 Physical Layer (PHY)

The PHY layer is responsible for the data service and the management of the data service using a certain radio channel according to a specific modulation and spreading techniques. The management part is the interface to the higher layers, and the data service part enables the transmission and reception of PHY protocol data units (PPDU) over the radio channel.

The IEEE 802.15.4 offers three operational (unlicensed) frequency bands: 2.4 GHz (worldwide, 16 channels), 915 MHz (North America and some Asian countries, 10 channels) and 866 MHz (Europe, 1 channel). The protocol also allows dynamic channel selection, a channel scan function in search of a beacon, receiver energy detection, link quality indication and channel switching. The data rate is 250 kbps at 2.4 GHz, 40 kbps at 915 MHz and 20 kbps at 868 MHz. In addition to these three frequency band patterns, two high data rate patterns have been added to the 868/915 MHz bands in the last revision of the standard IEEE 802.15.4REVb-2006. The higher data rates are achieved by using of the different modulation formats. This revision of the standard is backward-compatible to the IEEE 802.15.4-2003, meaning that devices conforming to IEEE 802.15.4REVb-2006 are capable of joining and functioning in a PAN composed of devices conforming to IEEE 802.15.4-2003. All of these frequency bands are based on the Direct Sequence Spread Spectrum (DSSS) spreading technique.

This thesis only considers the 2.4 GHz band with 250 kbps data rate, which is supported by the MICAz motes [8] used in the experimental test-beds.

5.4 IEEE 802.15.4 Medium Access Control (MAC)

The MAC sub-layer of the IEEE 802.15.4 protocol provides an interface between the physical layer and the higher layer protocols of LR-WPANs.

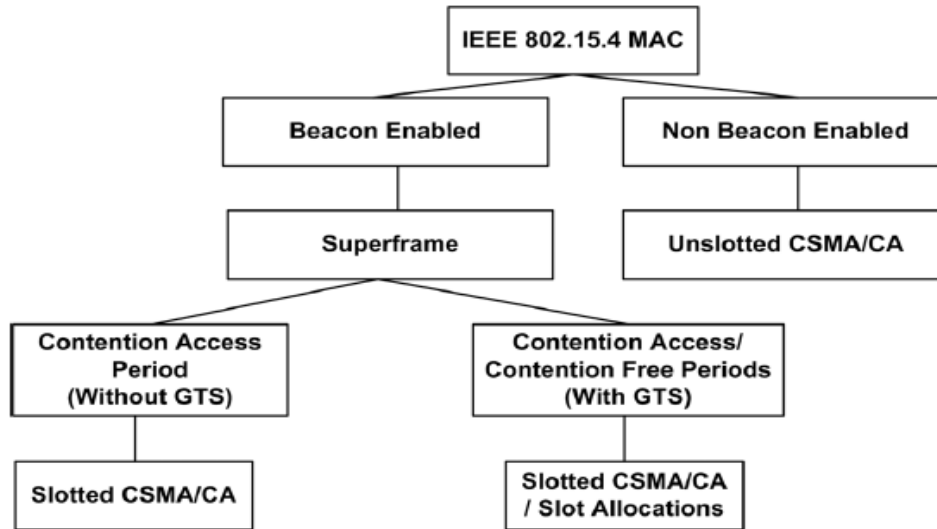


Figure 5.3: IEEE 802.15.4 Operational Modes

The MAC sub-layer of the IEEE 802.15.4 protocol has many common features with the MAC sub-layer of the IEEE 802.11 protocol, such as the use of CSMA/CA (*Carrier Sense Multiple Access / Contention Avoidance*) as a channel access protocol, the support of contention-free and contention-based periods. However, the specification of the IEEE 802.15.4 MAC sub-layer is adapted to the requirements of LR-WPAN as, for instance, eliminating the RTS/CTS mechanism (used in IEEE 802.11) to reduce the probability of collisions, since collisions are more likely to occur in low rate networks.

Figure 5.3 presents a structure of the IEEE 802.15.4 operational modes.

5.5 IEEE 802.15.4 Operational Modes

Following are two Operational modes in IEEE 802.15.4.

- ✂ **Non Beacon-enabled mode:** In non beacon-enabled mode (Figure 5.4(a)), the devices can simply send their data by using unslotted Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). There is no use of a superframe structure in this mode. The advantages of this mode are a scalability and self-organization. However, the non beacon-enabled mode cannot provide any time guarantees to deliver data frames. In fact, the "collision avoidance" mechanism is based on a random delay prior to transmission, which only reduces the probability of collisions. Thus, this mode cannot ensure collision-free and predictable access to the shared wireless medium and, consequently, it cannot provide any time and resource guarantees.

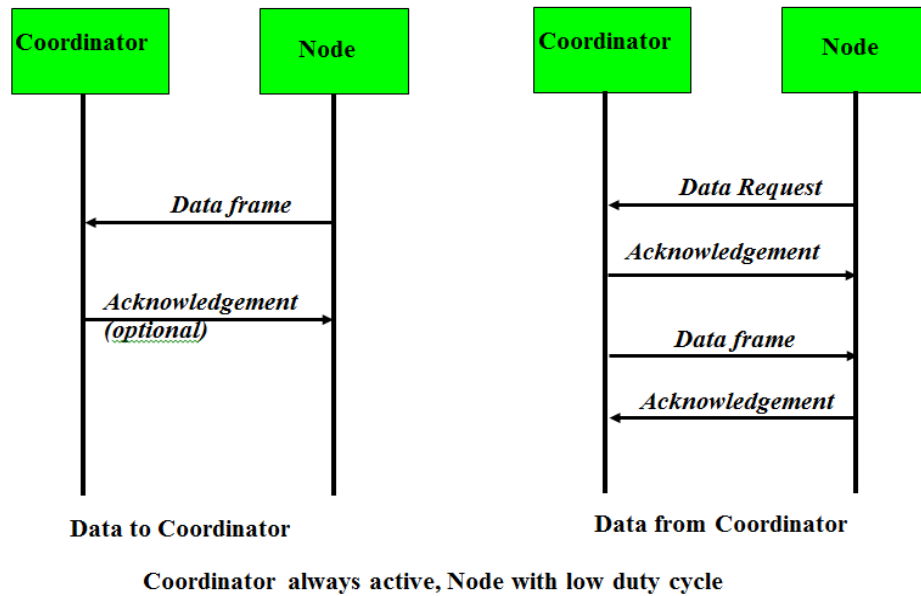


Figure 5.4 (a): Non Beacon mode

✂ **Beacon-enabled mode:** In beacon-enabled mode (Figure 5.4(b)), the beacon frames are periodically generated by the PAN Coordinator to identify its PAN, to synchronize nodes (i.e. coordinators or/and end devices) that are associated to it and to describe the structure of the superframe (Figure 5.5). It provides the energy conservation using low duty cycles, and the provision of collision-free and predictable access to the wireless medium through the Guaranteed Time Slot (GTS) mechanism. Thus, when the timeliness and energy efficiency are the main concerns, the beacon enabled mode should be employed. [9]

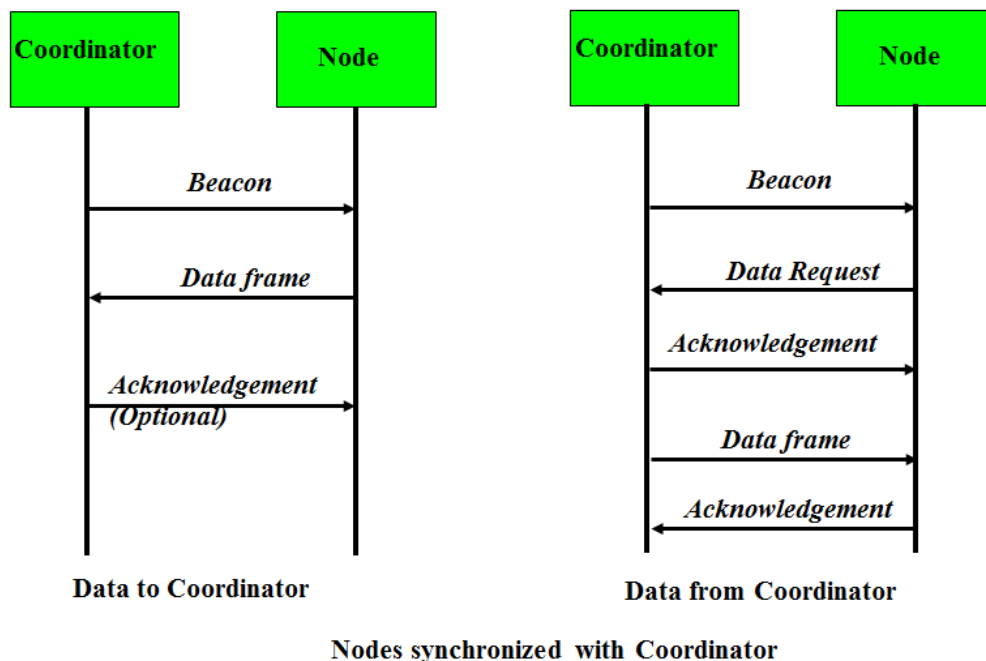


Figure 5.4 (b): Beacon mode

When the coordinator selects the beacon-enabled mode, it forces the use of a superframe structure to manage communication between devices (that are associated to that PAN). The format of the superframe is defined by the PAN coordinator and transmitted to other devices inside every beacon frame, which is broadcasted periodically by the PAN coordinator. The superframe is divided into 16 equally sized slots and is followed by a predefined inactive period. The superframe structure is discussed in section 5.6.



Figure 5.5: the Superframe Structure

As shown in Figure 5.5, the superframe is contained in a Beacon Interval, which is bounded by two consecutive beacon frames, and includes one *Contention- Access Period* (CAP) and may include also a *Contention-Free Period* (CFP), as outlined next:

- ✖ If communications are restricted to the CAP (defined in the beacon, issued by the PAN Coordinator) a device wishing to communicate must compete with other devices using a slotted CSMA/CA mechanism. All transmissions must be finished before the end of the superframe, i.e., before the beginning of the inactive period (if exists).
- ✖ If some guaranteed QoS is to be supported, then a *Contention-Free Period* (CFP) is defined. The CFP consists in *Guaranteed Time Slots* (GTSs) that may be allocated by the PAN coordinator to applications requiring low-latency or specific data bandwidth requirements. The CFP is a part of the superframe and starts at a slot boundary immediately following the CAP. The PAN coordinator may allocate up to seven GTSs and each GTS may occupy more than one time slot. With this superframe configuration, all contention-based communication must be finished before the start of the CFP, and a node transmitting a GTS must ensure that its transmission will be complete before the start of the next GTS (or the end of the CFP). According to the standard, the GTS is used only for communications between a PAN coordinator and a device. The GTS management are discussed in Section 6.7 in Chapter 6.

In both configurations (CAP only or CAP/CFP), the superframe structure can have an *inactive period* during which the PAN coordinator does not interact with its PAN and may enter in a low power mode. Switching the network between activity/inactivity periods is very suitable for devices where reduced energy consumption is a main concern. In fact, the inactive periods enable the devices to save energy and thus extend network lifetime.

5.6 The Superframe Structure

The superframe is contained in a Beacon Interval bounded by two beacon frames, and has an active period and an inactive period (see Figure 5.6).

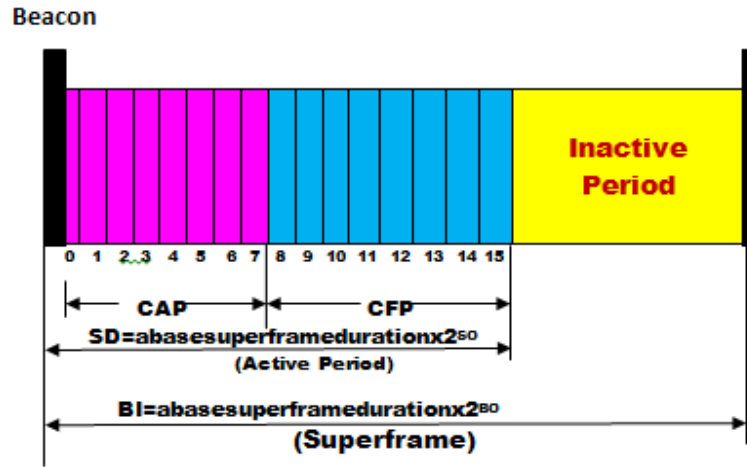


Figure 5.6: Example of the structure of a Superframe

The coordinator interacts with its PAN during the active period, and enters in a low power mode (sleep) during the inactive period. The structure of a superframe is defined by two parameters [7]:

- ✎ *macBeaconOrder* (BO): this attribute describes the interval at which the coordinator must transmit beacon frames. The value of the *macBeaconOrder* and the *Beacon Interval* (BI) are related as follows:

For $0 \leq BO \leq 14$,

$$BI = aBaseSuperframeDuration * 2^{BO} \text{ symbols}$$

- ✎ *macSuperframeOrder* (SO): this attribute describes the length of the active portion of the superframe, which includes the beacon frame. The value of the *macSuperframeOrder* and the *Superframe Duration* (SD) are related as follows:

For $0 \leq SO \leq BO \leq 14$

$$SD = aBaseSuperframeDuration * 2^{SO} \text{ symbols}$$

If $SO = BO \Rightarrow SD = BI$ and then the superframe is always active. According to the standard, if $SO = 15$, the superframe will not be active following the beacon. Moreover, if $BO = 15$, then the superframe shall not exist and the network will operate in the non beacon-enabled mode. In this case, the value of SO is ignored. As a result, a PAN that wishes to use the superframe structure must set *macBeaconOrder* to a value between 0 and 14 and *macSuperframeOrder* to a value between 0 and the value of *macBeaconOrder*. Otherwise, the PAN will operate in a non beacon-enabled mode with a value of *macBeaconOrder* and *macSuperframeOrder* equal to 15. The active portion of each superframe is divided into $aNumSuperframeSlots = 16$ equally spaced slots of duration $2^{SO} * aBaseSlotDuration$. The attribute *aBaseSlotDuration* represents the number of symbols forming a superframe slot when the superframe order is equal to zero. The value of *aBaseSlotDuration* is equal to 60 symbols.

The active portion of the superframe structure is composed of three parts:

- ✖ **Beacon:** the beacon is transmitted without the use of CSMA at the start of slot 0. It contains the information on the addressing fields, the superframe specification, the GTS fields, the pending address fields, etc.
- ✖ **CAP:** the CAP starts immediately after the beacon frame and ends before the beginning of the CFP (if it exists). Otherwise, the CAP ends at the end of the active part of the superframe. The minimum length of the CAP is fixed at $aMinCAPLength = 440 \text{ Symbols}$. This minimum length ensures that MAC commands can still be transferred to devices when GTSs are being used. A temporary violation of this minimum may be allowed if additional space is needed to temporarily accommodate the increase in the beacon frame length needed to perform GTS management. All the transmissions during the CAP are made using a slotted CSMA/CA mechanism to access the channel. However, the acknowledgement frames and any data that immediately follows the acknowledgement of a data request command are transmitted without contention. A device that cannot complete its transmission one *Inter Frame Spacing* period before the end of the CAP, must defer its transmission until the CAP of the next superframe.
- ✖ **CFP:** The CFP starts immediately after the end of the CAP and must complete before the start of the next beacon frame. All the GTSs that may be allocated by the PAN coordinator are located in the CFP and must occupy contiguous slots.

The CFP may therefore grow or shrink depending on the total length of all GTSs. The transmissions in the CFP are contention-free and therefore do not use a CSMA/CA mechanism to access the channel. Additionally, a frame may only be transmitted if the transmission ends one IFS before the end of the correspondent GTS.

The IFS period defines the amount of time that separates the transmission of two consecutive frames. In fact, the MAC sub-layer needs a finite amount of time to process data received by the physical layer. In an acknowledged transmission the IFS follows the acknowledgement frame, otherwise the IFS follows the frame itself. The length of an IFS frame depends on the frame size. The transmission of short frames, whose sizes are lower than $aMaxSIFSFrameSize = 18 \text{ Bytes}$, is followed by a *SIFS* period of a duration of at least $aMinSIFSPeriod = 12 \text{ symbols}$. On the other hand, the transmissions of long frame, whose lengths are greater than $aMaxSIFSFrameSize$ is followed by a LIFS of duration of at least $aMinLIFSPeriod = 40 \text{ symbols}$.

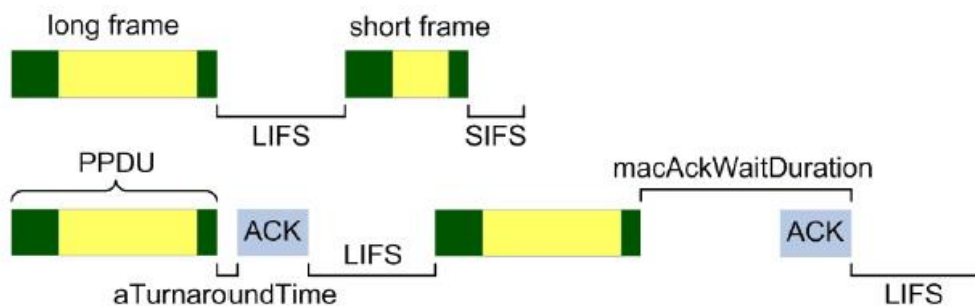


Figure 5.7: The Inter Framing Spacing

Figure 5.7 illustrates these concepts. The CSMA/CA must take this requirement into account for transmissions in the CAP.

5.7 Simulation Setup & Results¹

In this section, an accurate simulation model is provided with respect to the specifications of IEEE 802.15.4 standard in non-beacon mode as discussed in previous section. Two different scenarios viz. tree and mesh are simulated and analysed, where the topological features and performance of the IEEE 802.15.4 standard using OPNET simulator are examined. The comparative results have been reported for the performance metrics viz. Number of hops, End to End Delay, and Load of network [10].

¹ Published a paper: Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma, "Analytical Approach for Performance of Wireless Sensor Networks", *International Journal of Electronics and Computer Science Engineering*, 1877, Available Online at www.ijecse.org ISSN- 2277-1956.

- ❖ **Number of hops:** The number of hops is the number of times a packet travels from the source through the intermediate nodes to reach the destination.
- ❖ **End to End Delay (ETE):** This statistics represents the total delay (in seconds) occurs for the transmitted packet from source to the destination.
- ❖ **Throughput:** Throughput (bits/sec) is the average number of bits or packets successfully received or transmitted by the receiver or transmitter channel per second.
- ❖ **Load per PAN:** This statistics represents the total load (in bits/sec) for a particular PAN submitted to 802.15.4 MAC by all higher layers in all WPAN nodes of the network.
- ❖ **PAN Affiliation for Coordinator:** This represents time that the node joins a network.

✂ SCENARIO - 1

Tree Routing and Mesh Routing networks are designed and both are identical networks only difference is to configure PAN of Tree Routing with Default tree Network and PAN in Mesh Routing with Default Mesh Network. The same network tree structure forms in each case. The majority of the nodes have been configured with Random traffic; however Router 1 has been explicitly configured to send traffic to Router 3.

The PAN in the Tree Routing scenario has been configured as a Default Tree Network. Application traffic will be routed to the destination along the parent-child links of the network tree. The PAN in the Mesh Routing scenario has been configured as a Default Mesh Network. After mesh routes have been established (a few seconds after network formation), application traffic will be routed through the shortest possible route using any of the router or coordinator presents in the network. End devices do not participate in mesh routing, therefore they must still route traffic through their parent node.

Figure 5.8 shows that the number of hops taken by Router 1 to reaches its destination. This statistic for Tree Routing is of red line and for Mesh Routing is of blue line. Note that both lines begin at two hops, but the Mesh Routing line quickly changes to one hop for the remainder of the simulation.

In Figure 5.9, the red line indicates ETE Delay for the Tree Routing scenario while the blue line indicates ETE Delay for Mesh Routing. The ETE Delay for Mesh Routing is lower due to the mesh routing process finding more efficient routes than tree

based routing for some of the traffic. For some nodes, the tree based route will be the most efficient route, resulting in only a minor overall improvement in ETE Delay.

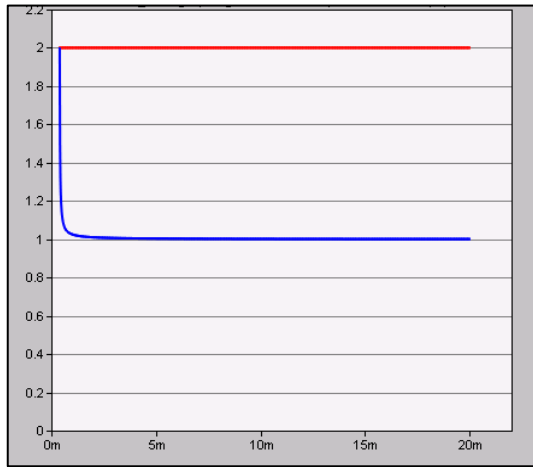


Figure 5.8: Number of Hops

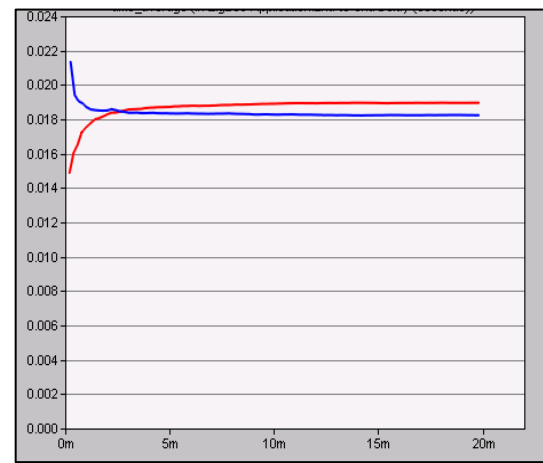


Figure 5.9: End to end delay(seconds)

In Figure 5.10 the red line indicates ETE Delay for Router 1 in the Tree Routing scenario while the blue line indicates ETE Delay for Router 1 in Mesh Routing. The ETE Delay in Mesh Routing is lower due to the mesh routing process finding more efficient route than tree based routing (1 hop vs. 2 hops).

The red line in Figure 5.11 is the total load for the Tree Routing scenario while the blue line is total load for Mesh Routing. The load for Mesh Routing is lower due to fewer hops for application traffic resulting in less overall traffic seen at the MAC layer. Also note that there is a very small spike in load for Mesh Routing near the beginning of the simulation that is not seen for Tree Routing. This is due to the routing messages being broadcast at that time.

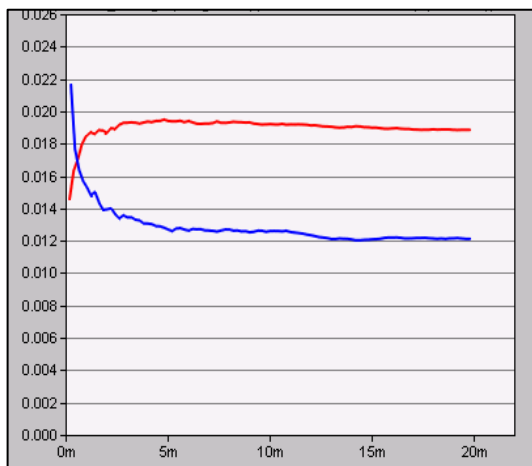


Figure 5.10: End to end delay(sec) for Router 1

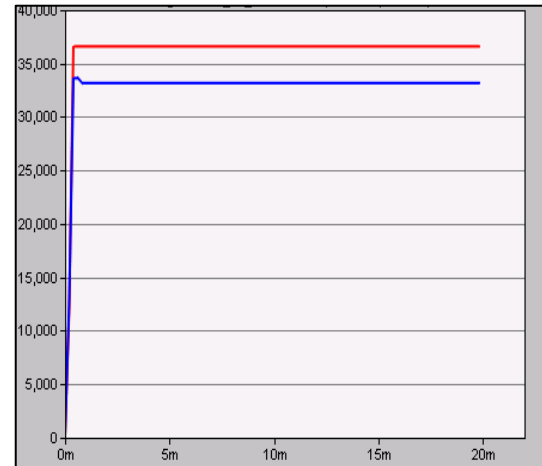


Figure 5.11: Load per PAN (bits/sec)

Figure 5.12 shows the throughput for the tree and mesh topologies respectively. It shows that the maximum throughput is achieved in Tree topology and Mesh topology has the low throughput. The reason for this is because Tree topology is communicating on the basis of the PAN coordinators and R which are more efficient as compared to the end devices. Also in Tree topology total load of the network is divided among the local PAN and Rs (Routers) as a result of which lesser collisions and lesser packet drops takes place as a result of which the throughput is maximum in case of Tree topology.

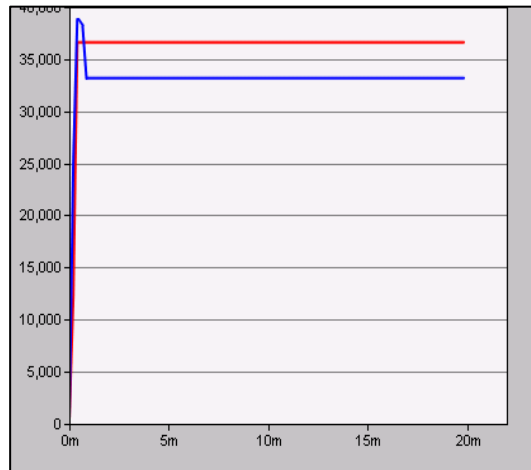


Figure 5.12 Throughput (bits/sec)

Figure 5.13(a) and (b) shows the tree structure and mesh route from Router 1. The tree structure in this scenario is identical to the previous one. Also note that the tree path from Router 1 to Router 3 passes through the Coordinator. In Mesh Network, the mesh route goes directly to Router 3.

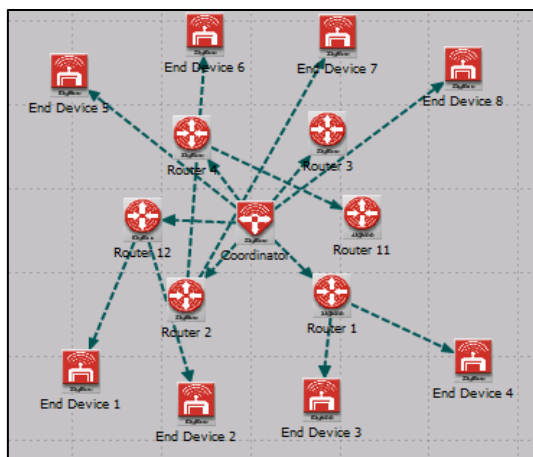


Figure 5.13(a) : Tree structure

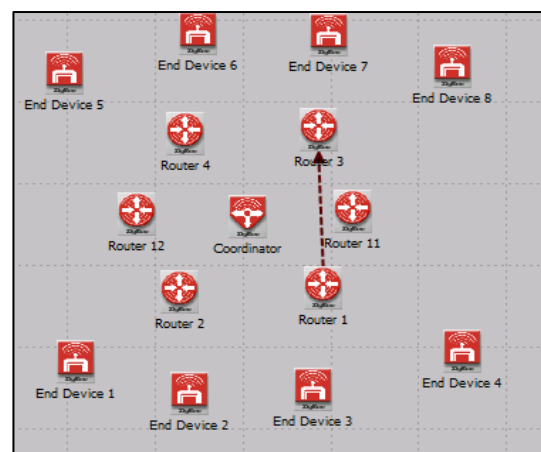


Figure 5.13(b) : Mesh Route

✂ SCENARIO - 2

Figure 5.14 scenario studies the behavior of a network when the coordinator fails. The network contains two coordinators and 24 routers and end devices. Each router and end device in the scenario has its PAN ID set to Auto-Assigned. The coordinators have their PAN ID's set to 1 and 2 respectively. A portion of the remaining nodes should join each of the two coordinators (any given node may join either coordinator).

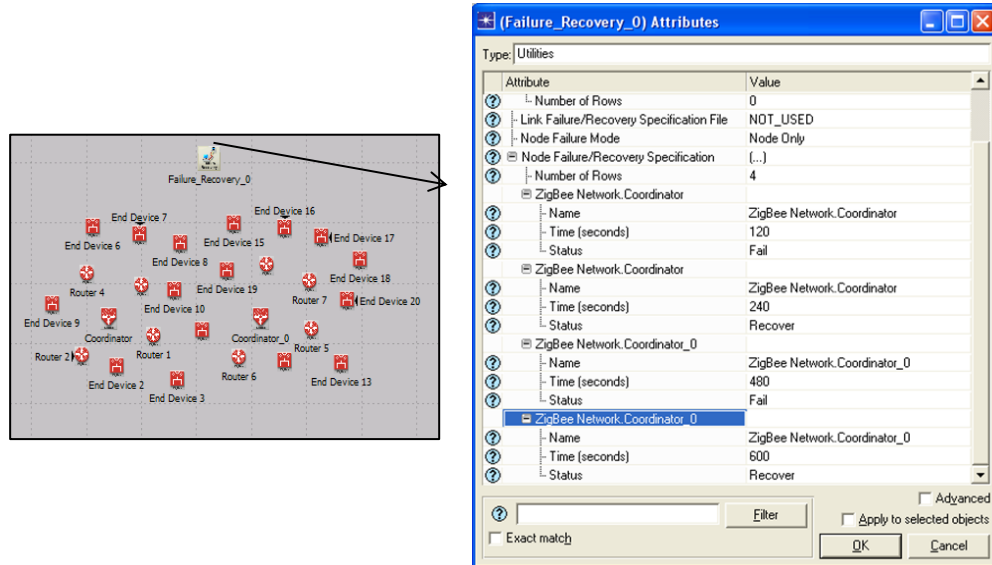


Figure 5.14 : Snapshot of the Network when Coordinator Fails

Two minutes into the simulation, the first coordinator fails. It will remain failed until four minutes, when it will recover and re-establish a network. At eight minutes, the second coordinator will fail. It will remain failed until ten minutes. The expected behavior is approximately half the nodes to join each of the two coordinators at initialization. When the first coordinator fails, the nodes joined to that PAN should leave and join the second coordinator. When the second coordinator fails, all the nodes should join the first coordinator.

Figure 5.15 shows Global MAC load for PAN. The blue line indicates the first coordinator (PAN 1) while the red is the second coordinator (PAN 2). Initially, both PANs have more or less equivalent loads (PAN 1's is greater due to a few more nodes joining that network). After 2 minutes, PAN 1's load drops to zero while PAN 2's load increases. Note that these numbers remain constant even after the first coordinator recovers at 4 minutes. When the second coordinator fails after eight minutes, PAN1's load increases and PAN 2's falls to zero at that time.

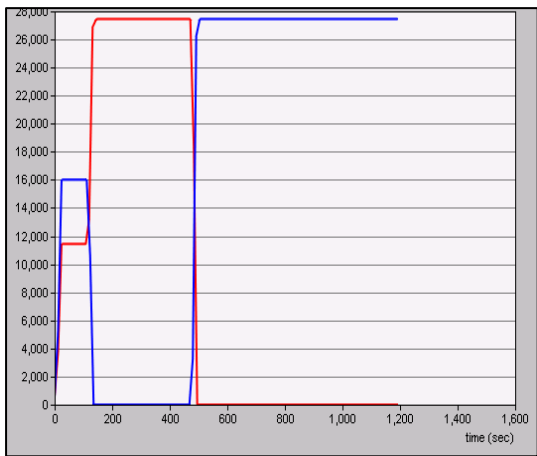


Figure 5.15: Load per PAN (bits/sec)

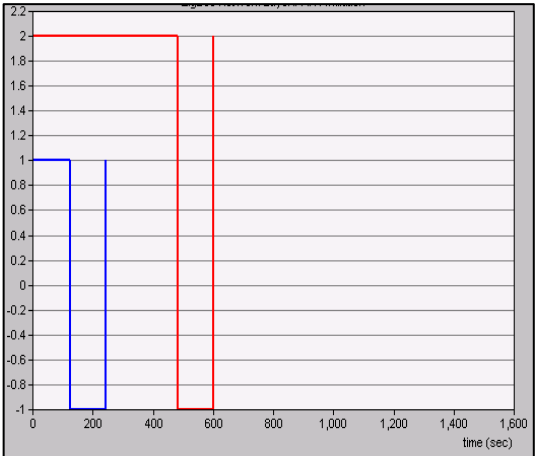


Figure 5.16 : PAN Affiliation

Figure 5.16 shows the PAN affiliation for coordinator and coordinator_0 in the network. From these two graphs you can see that during the their failed periods the coordinators have a PAN ID of -1, which is the code indicating they are not currently joined to the network.

	PAN ID	Number of Nodes	Tree Depth
1	1	15	2
2	2	11	2

	PAN ID	Number of Nodes	Tree Depth
1	2	25	3
2	1	1	0

	PAN ID	Number of Nodes	Tree Depth
1	1	25	3
2	2	1	0

Figure 5.17: Global Output Report at simulation time 360, 60,720

Figure 5.17 shows the global Output report derived from the simulation. From the Number of Nodes column in the three tables presented here, initially 15 nodes are joined to PAN 1 while 11 are joined to PAN 2 (roughly half each, with an acceptable variation due to randomness). After the first failure, 25 nodes are joined to PAN 2 while only one (the coordinator itself) is joined to PAN 1. After the second failure, the reverse is true.

Summary


In this chapter, the structure IEEE 802.15.4, widely used for WSN is discussed in terms of its operational modes. The Superframe Structure has been elaborated to manage communication between the devices of the WSN. Parametric evaluation for Topological structure of IEEE 802.15.4 has been carried out and analyzed.



Chapter 6



IEEE 802.15.4 WSN : GTS Mechanism



This Chapter details about the performance of the GTS allocation mechanism in IEEE 802.15.4. The analysis gives a full understanding of the behavior of the GTS mechanism with regards to Packet medium access delay, throughput and wasted bandwidth due to IFS. The impact of IEEE 802.15.4 parameters on the throughput, delay of a GTS allocation are analyzed which pave the way for an efficient dimensioning of an IEEE 802.15.4 cluster.

With the emergence of new Wireless Sensor Network (WSN) applications under timing constraints, the provision of deterministic guarantees may be more crucial than saving energy during critical situations. The IEEE 802.15.4 protocol [1] is one potential candidate to achieve predictable real-time performance for Low-Rate Wireless Personal Area Networks (LR-WPAN).

Timeliness is an important feature of the IEEE 802.15.4 protocol, turning it quite appealing for applications under timing constraints. Because of this attractive feature, it is used in real time for time constraint data delivery which is provided by Guaranteed Time slot mechanism.

6.1 GTS Mechanism Evaluation

The goal for any simulation model is to accurately model and predict the behaviour of a real system. Recently, several analytical and simulation models of the IEEE 802.15.4 [1] protocol have been proposed. Nevertheless, currently available simulation models [2] for this protocol are both inaccurate and incomplete, and in particular they do not support the Guaranteed Time Slot (GTS) mechanism, which is required for time-sensitive wireless sensor applications.

6.2 Related Work

NS-2, OMNeT++ and Opnet are widely used and popular network simulators which, among others, include a simulation model of the IEEE 802.15.4 protocol. Of course, each simulator has its own disadvantages and advantages, which is already discussed in chapter 3. The Network Simulator 2 (ns-2) [3] is an object-oriented discrete event simulator including a simulation model of the IEEE 802.15.4 protocol. The accuracy of its simulation results is questionable since the MAC protocols, packet formats, and energy models are very different from those used in real WSNs [4]. This basically results from the facts that ns-2 was originally developed for IP-based networks and further extended for wireless networks. Moreover, the GTS mechanism was not implemented in the ns-2 model. OMNeT++ (Objective Modular Network Test-bed in C++) [5] is another discrete event network simulator supporting unslotted IEEE 802.15.4 CSMA/CA MAC protocol only. Finally, note that while ns-2 and OMNeT++ are open

source projects, the Opnet Modeler is commercial project providing a free of charge university program for academic research projects [6].

There have also been several research works on the performance evaluation of the IEEE 802.15.4 protocol using simulation model. Author in [7] compared the beacon enabled and non-beacon enabled modes with and without acknowledgement on OPNET simulator to analyse the loading effect on considered scenario by varying number of nodes. Author in [8] underline the suitability of 802.15.4 in wireless medical applications used in patient care applications through the network performance evaluation. Their focus is on interoperability and scalability. J. Zheng and M.J. Lee in [9] implemented the IEEE 802.15.4 standard on NS2 simulator and provided simulation-based performance evaluation on 802.15.4. It was a comprehensive literature that defines the 802.15.4 protocol and was mainly confined to the IEEE 802.15.4 MAC performances. This work has a minor evaluation on the performance of the peer-to-peer networks [10]. In [11] the authors have proposed an accurate OPNET simulation model, with focus on the implementation of the GTS mechanism. Based on the simulation model they proposed a methodology to tune the protocol parameters to guarantee better performance of the protocol by maximizing the throughput of the allocated GTS as well as minimizing frame delay.

Our work here focuses on simple 1-hop star network. It describes the wireless sensor networks in the IEEE802.15.4 to integrate the GTS mechanism in the MAC layer in order to improve the QoS. To achieve this performance evaluation, a simulation model for the IEEE 802.15.4 GTS mechanism within the implementation of the protocol under the OPNET simulator is used [11]. This simulation model has been recently made available publicly [12]. OPNET Modeler was chosen due to its accuracy and to its sophisticated graphical user interface. This model is used to carry out a set of experiments. The Focus is on extending this simulation paradigm by introducing additional settings and performance metrics.

6.3 The IEEE 802.15.4 Simulation Model

The simulation model implements physical and medium access control layers defined in the IEEE 802.15.4-2003 standard [1]. The OPNET Modeler [11] is used due to its accuracy and to its sophisticated graphical user interface. The OPNET Modeler is an industry leading discrete-event network modeling and simulation environment. The wireless module extends the functionality of the OPNET Modeler with accurate modeling, simulation and analysis of wireless networks. This module is one of the several

add-on modules available from OPNET. The actual version of the simulation model only supports the **star topology** where the communication is established between devices, called inside the model *End Devices*, and a single central controller, called *PAN Coordinator*. Each device operates in the network must have a unique address. The structure of the IEEE 802.15.4 simulation model is presented in Figure 6.1. The structure of the IEEE 802.15.4 sensor nodes (*wpan_sensor_node*) used in the simulation model is composed of four functional blocks [13]:

1. The **Physical Layer** consists of a wireless radio transmitter (*tx*) and receiver (*rx*) compliant to the IEEE 802.15.4 specification, operating at the 2.4 GHz frequency band and a data rate equal to 250 kbps. The transmission power is set to 1 mW and the modulation technique is Quadrature Phase Shift Keying (QPSK).
2. The **MAC Layer** implements the slotted CSMA/CA and GTS mechanisms. The GTS data traffic (i.e. time-critical traffic) incoming from the application layer is stored in a buffer with a specified capacity and dispatched to the network when the corresponding GTS is active. The non time-critical data frames are stored in an unbounded buffer and based on the slotted CSMA/CA algorithm are transmitted to the network during the active CAP. This layer is also responsible for generating beacon frames and synchronizing the network when a given node acts as PAN Coordinator.
3. The **Application Layer** consists of two data traffic generators (i.e. *Traffic Source* and *GTS Traffic Source*) and one *Traffic Sink*. The Traffic Source generates unacknowledged and acknowledged data frames transmitted during the CAP (uses slotted CSMA/CA). The GTS Traffic Source can produce unacknowledged or acknowledged time-critical data frames using the GTS mechanism. The Traffic Sink module receives frames forwarded from lower layers and performs the network statistics.
4. The **Battery Module** computes the consumed and the remaining energy levels. The default values of the current draws are set to those of the MICAz mote specification [14].

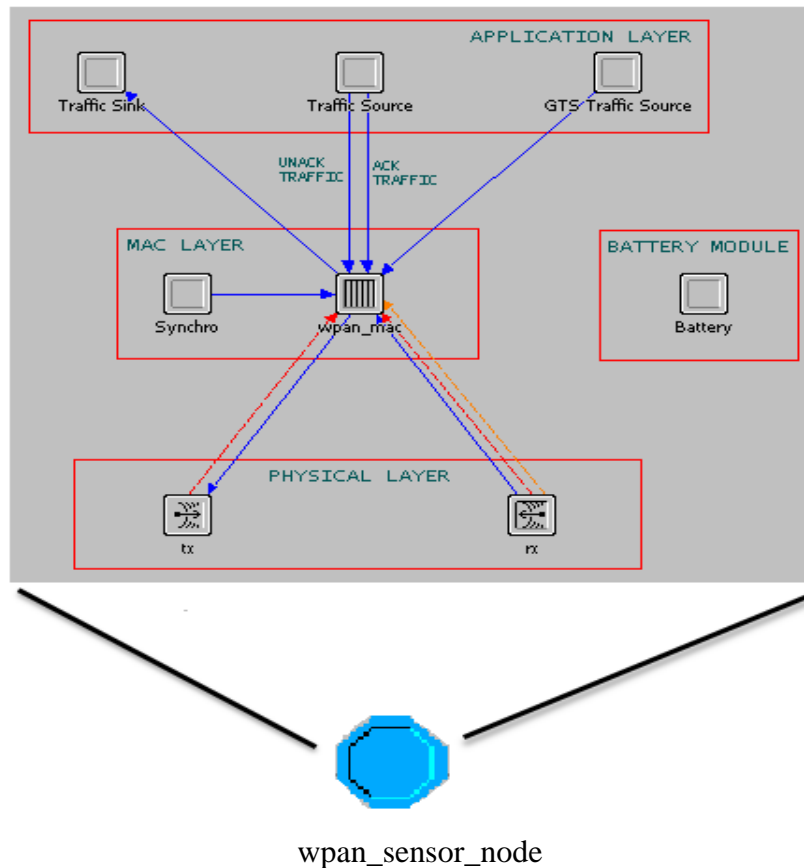


Figure 6.1: The structure of the IEEE 802.15.4 Simulation Model

The actual version of the simulation model is 2.0 and is not backward-compatible to the previous version 1.0, meaning that the devices conforming to version 1.0 are not capable of joining and functioning in a PAN composed of devices conforming to version 2.0 and vice-versa. The actual version 2.0 of the simulation model implements the following functions in accordance with the IEEE 802.15.4-2003 standard.

Support (implemented) features:

- ✘ Beacon-enabled mode
- ✘ Slotted CSMA/CA MAC protocol
- ✘ Frame formats (beacon, command, ack, mac_packet)
- ✘ Physical layer characteristics
- ✘ Computation of the power consumption (MICAz and TelosB (TmoteSky) motes supported) - Battery Module
- ✘ Guaranteed Time Slot (GTS) mechanism (GTS allocation, deallocation and reallocation functions)

- ✖ Generation of the acknowledged and unacknowledged application data (MAC Frame payload = MSDU) transmitted during the Contention Access period (CAP)
- ✖ Generation of the acknowledged or unacknowledged application data transmitted during the Contention Free Period (CFP)

The values of all constants and variables in this simulation model are considered for the 2.4 GHz frequency band width, a data rate of 250 kbps, which is supported by the MICAz or TelosB motes, for example. In this case, one symbol corresponds to 4 bits. For other frequency bands and data rates it is necessary to change appropriate parameters inside the simulation model (e.g. the header file *wpan_params.h*).

There are two types of nodes inside the simulation model:

- ✖ ***wpan_analyzer_node***: This node captures global statistical data from whole PAN (one within PAN).
- ✖ ***wpan_sensor_node***: This node implements the IEEE 802.15.4-2003 standard as was mentioned above.

6.4 GTS Mechanism

In Chapter 5, it is discussed that in IEEE 802.15.4 standard, Superframe is divided into active and optional inactive period. Each active period can be further divided into a Contention Access Period (CAP) and an optional Contention Free Period (CFP). The CFP is activated by the request sent from an End Device to the PAN Coordinator and provides real-time guarantees for time-critical data. Upon receiving this request, the PAN Coordinator checks whether there are sufficient resources and, if possible, allocates the requested time slots. This requested group of time slots is called Guaranteed Time Slot (GTS) and is dedicated exclusively to a given device. A CFP support up to 7 GTSs and each GTS may contain multiple time slots. The GTS allows the corresponding device to directly access the medium without contention with other devices inside the PAN. A GTS can only be allocated by the PAN coordinator and applied exclusively to data transfer between the device and its coordinator, either from the End Device to the PAN Coordinator (transmit direction) or from the PAN Coordinator to the End Device (receive direction). Each device may request one GTS in the transmit direction and/or one GTS in the receive direction. The allocation of the GTS cannot reduce the length of the CAP to less than *aMinCAPLength* (440 symbols). If the available resources are not sufficient, the

GTS allocation is denied by the PAN Coordinator. End Device to which a GTS has been allocated can also transmit during the CAP.

The PAN Coordinator may accept or reject the GTS allocation request from the End Device according to the value of the user defined attribute ***GTS Permit***. The End Device can specify the time when the GTS allocation and deallocation requests are sent to the PAN Coordinator (***Start Time*** and ***Stop Time*** attributes). This allocation request also includes the number of required time slots (***GTS Length*** attribute) and their direction, transmit or receive (***GTS Direction*** attribute).

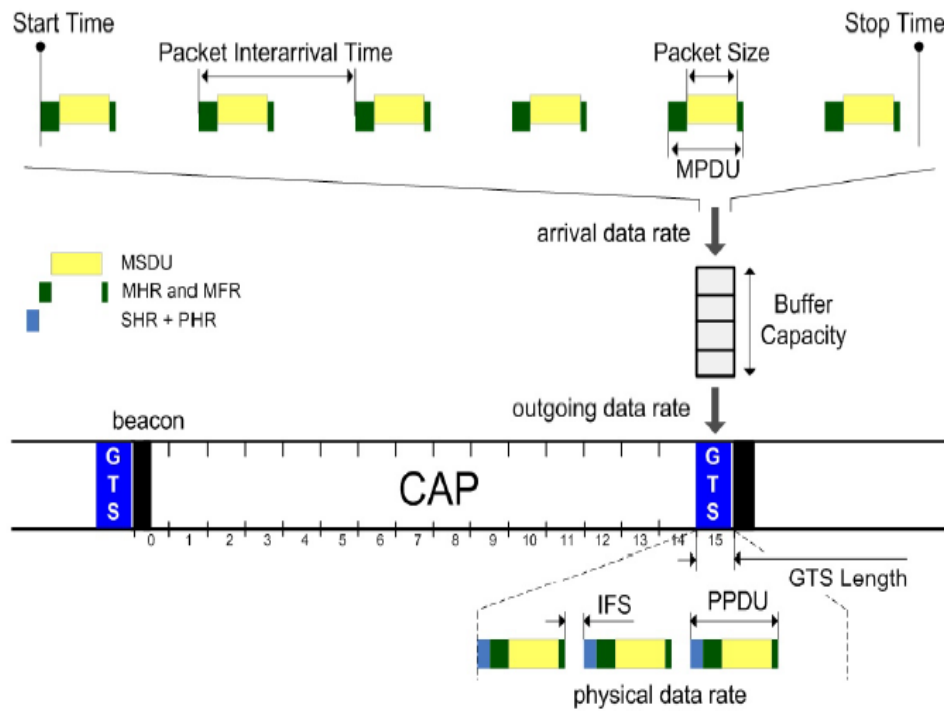


Figure 6.2 : Packet flow structure in GTS enabled mode [15]

The GTS mechanism packet flow structure is shown in Figure 6.2. When the requested GTS is assigned to a given device, its application layer starts generating data blocks that correspond to the MAC frame payload (i.e. MAC Service Data Unit (MSDU)). The size of the frame payload is specified by the probability distribution function of the ***MSDU Size*** attribute. The probability distribution function, specified in the ***MSDU Interarrival Time*** attribute, defines the inter-arrival time between two consecutive frame payloads. Then, the frame payload is wrapped in the MAC header and stored as a frame in the buffer with a given capacity (***Buffer Capacity*** attribute). The default size of the MAC header (***MAC_HEADER_SIZE***) is 104 bits, since only 16-bit

short addresses are used for communication (according to standard specification). The maximum allowed size of the overall frame (i.e. frame payload plus the MAC header) is equal to $aMaxPHYPacketSize$ (1016 bits). The generated frames exceeding the buffer capacity are dropped. When the requested GTS is active, the frames are removed from the buffer, wrapped in the PHY headers and dispatched to the network with an outgoing data rate equal to physical data rate $WPAN_DATA_RATE$ (250 kbps).

Consecutive frames are separated by inter-frame spacing (IFS) periods. The IFS is the minimum interval that any device must wait before sending another frame. Determination of IFS is depicted in Figure 6.3. The IFS is equal to a short inter-frame spacing (SIFS) of 48 bits, for frame lengths smaller than $aMaxSIFSFrameSize$ (144 bits). Otherwise, the IFS is equal to long inter-frame spacing (LIFS) of 160 bits, for a frame length greater than $aMaxSIFSFrameSize$ bits and smaller than $aMaxPHYPacketSize$ (1,016 bits) [1].

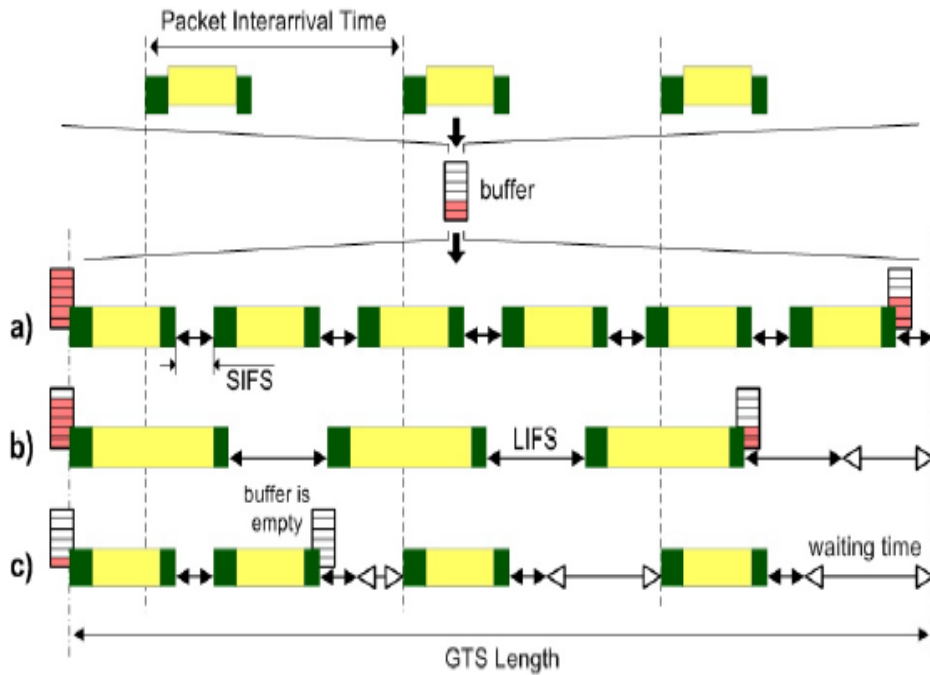


Figure 6.3: The utilization of the transmission time inside the GTS [15]

The MAC header in this model is 104 bits. To analyse the impact of IFS on the GTS performance, random packet sizes were determined at the input; 40 bits, 65 bits, 258 bits, 724 bits, 811 bits and 912 bits. The 40 bits and 65 bits packet sizes were chosen to invoke the SIFS, whereas the others will use LIFS.

Following Table 6.1 and 6.2 show the description of user defined attributes.

Attribute name	Value	Description
GTS Permit	[enabled/disabled]	Enabled if the PAN Coordinator accepts GTS request from End Device. Disabled otherwise.
Start Time	Sec	The absolute simulation time in seconds when the application layer will start its GTS data generation. Setting the value to Infinity will simply disable data generation.
Stop Time	Sec	The absolute simulation time in seconds when the application layer will stop its GTS data generation. Setting the value to Infinity will make the application layer generate GTS data until the end of the simulation.
Length	0-15 slots	The length of the GTS in superframe slots within one superframe.
Direction	Transmit/receive	The direction of the transmission from the End Device point of view: <ul style="list-style-type: none"> ➤ transmit: End Device → PAN Coordinator ➤ receive: End Device ← PAN Coordinator
Buffer Capacity	Bits	The capacity of the FIFO buffer for storing the traffic arriving from the application layer before dispatching to the network.

Table 6.1 GTS setting

Attribute name	Value	Description
MSDU Inter arrival Time	Sec	The inter-arrival time in seconds between two consecutive MSDUs.
MSDU size	Bits	The size in bits of the generated data unit i.e. MSDU. The value can be from the interval (0, aMaxMACFrameSize). The values out of this interval are automatically set to boundary values. aMaxMACFrameSize is the maximum size that can be transmitted in the MAC Frame Payload and is equal to aMaxPHYPacketSize - MAC_HEADER_SIZE (i.e. 1016 - 104 = 912 bits)
Acknowledgement	Disabled/enabled	Enable if the generated data should be acknowledged.

Table 6.2 GTS Traffic Parameters

6.5 Parameter Analysis

To evaluate GTS Mechanism following parameters are consider: GTS Throughput, Packet Medium Access Delay and Wasted Bandwidth.

6.5.1 Throughput Analysis:

Defining the maximum throughput of GTS [16] is the maximum bandwidth for data transmission, the guaranteed bandwidth of a GTS is, (Eq1).

$$R = \frac{2^{IO}C}{16} - \frac{T_{idle}}{BI} C = \lambda \cdot DC \cdot C - W_{idle} \quad \text{----- (1)}$$

With $\lambda = 1/16$,

$DC = \text{duty cycle} = 2^{IO} \leq 1$ where $IO=SO-BO$

$C = 250 \text{ kbps}$,

$$W_{idle} = W_{overhead} + W_{wasted}$$

The maximum throughput can be expressed as follows:

$$Th_{\max} = \frac{T_{data}}{BI} \cdot C \quad T_{data} = T_s - T_{idle} \quad \text{----- (2)}$$

$$T_s = \frac{SD}{16} = \text{abaseSuperframeDuration} * 2^{(SO-4)} \quad \text{----- (3)}$$

$$T_{idle} = T_{overhead} + T_{wasted} \quad \text{----- (4)}$$

T_{idle} is the sum of idle time spent inside a GTS

$$T_{overhead} = T_{IFS} + T_{ACK} \cdot \prod ACK \quad \text{----- (5)}$$

T_{IFS} is the frame interval time when the data frame transmission wasted. T_{ACK} is time when wait for Acknowledgment frame, where $\prod ACK = 1$ for an acknowledged transaction and $\prod ACK = 0$ for an unacknowledged transaction. The value T_{wasted} is greater than zero if the length of a GTS is longer than the transaction time.

Hence, for a given SO the maximum throughput is related to the maximum time effectively used for data transmission inside a GTS. The transmission frame interval is affected by the arrival of data frames, T_{wasted} is affected by the size of buffer capacity. If data is small compared to GTS slot size, throughput decreases due to wastage of remaining bandwidth. So it is necessary to design the Superframe Structure in such a way that wasted bandwidth is reduced. This can be an important factor only for WSN application where a small data is to be transmitted at a regular interval of time, sometimes only signal that contained data in terms of bit.

6.6 Simulation Setup

The Simulation setup consists of a simple star network containing one pan coordinator and one associated end device which is GTS enabled. This configuration is sufficient for the performance evaluation of the GTS mechanism, since there is no medium access contention [17]. Thus, having additional devices would have no influence

on the simulation results. Simulations for two different SO and BO values (varying duty cycles) are run to thoroughly study the performance impact.

For the sake of simplicity, and without loss of generality, assume the allocation of only one time slot GTS in transmit direction. The first simulation considers 100% duty cycle (i.e. $SO = BO$). In what follows, the change of the SO means that the BO also changes while satisfying $SO = BO$. This means that the optional inactive portion is not included in the superframe.

The statistical data is computed from a set of 1,000 samples. For the GTS allocation mechanism, the simulation time of one run is equal to the duration of 1,000 superframe periods, and consequently the simulation time depends on the SO. The 912 bits is the maximum allowable bits not to exceed aMaxPHYPacketSize (1,016) for 1 frame.

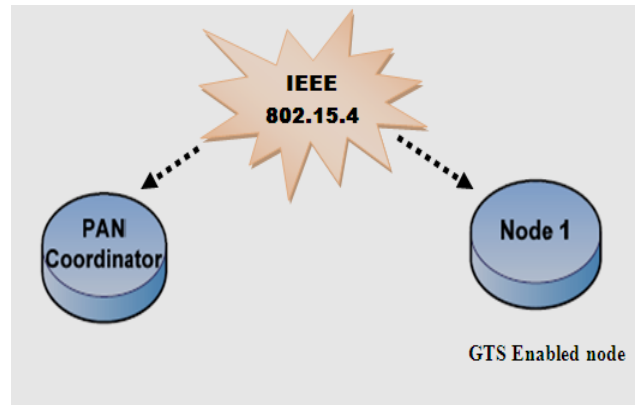


Figure 6.4: Simulation model

Figure 6.4 shows the implemented GTS Simulation model. For the parameter evaluation of the GTS Mechanism, GTS is set to 1 time slot and acknowledgement traffic is disabled. The buffer capacity is then dimensioned to 1 Kbits and 4 Kbits for each simulation and simulation time is 10 sec. 4 kbits is the real buffer size in MicaZ [14] architecture. Table 6.3 shows the attributes taken under simulation consideration.

Parameters	Values	Units
Packet size	40,65,258,724,811,912	Bits (constant type)
Packet Interarrival Time	0.001824	Sec
Buffer size	1 K, 4 K	Bits (constant type)
Distance	25	Meters
Transmission band	2.4	GHz
Transmission Power	1	Miliwatt

TABLE 6.3 SIMULATION PARAMETERS

6.7 Simulation Results and Discussion

This section evaluates the impact of SO, duty cycle, the inter-arrival time, the buffer capacity and the packet size on the data throughput of the allocated GTS, Packet Medium Access Delay, Wasted Bandwidth of the transmitted GTS frames.

6.7.1 Impact of Buffer size on GTS Throughput ¹

By varying the buffer size for 1 K and 4 K, the GTS throughput is observed [18]. From the Figure 6.5(a) and (b), it is observed that 912 packet size fails to transmit in all SO when the buffer size is 1 Kbits. This is because the 1 Kbits buffer capacity is not sufficient to hold the packet size. For lower SO values, superframe is very small so only 40 bits packet size, we get the GTS throughput. Therefore, 40 bits can conform for small SO values.

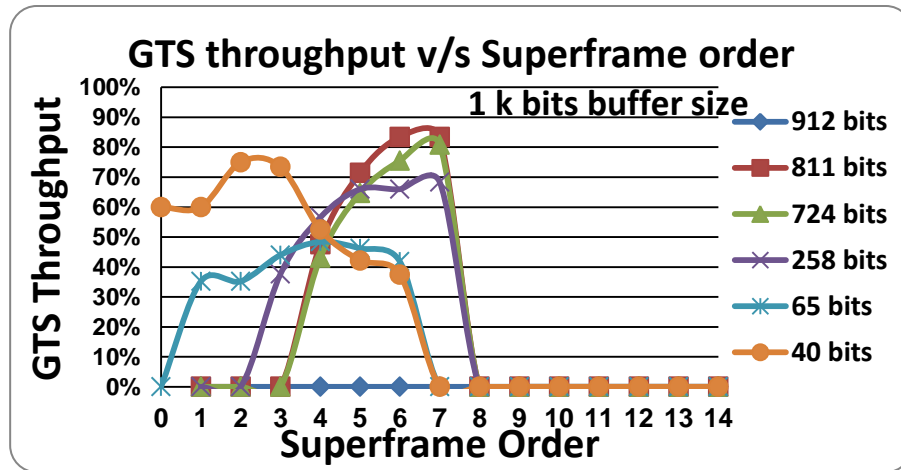


Figure 6.5(a): GTS throughput v/s superframe order for 1 k buffer capacity

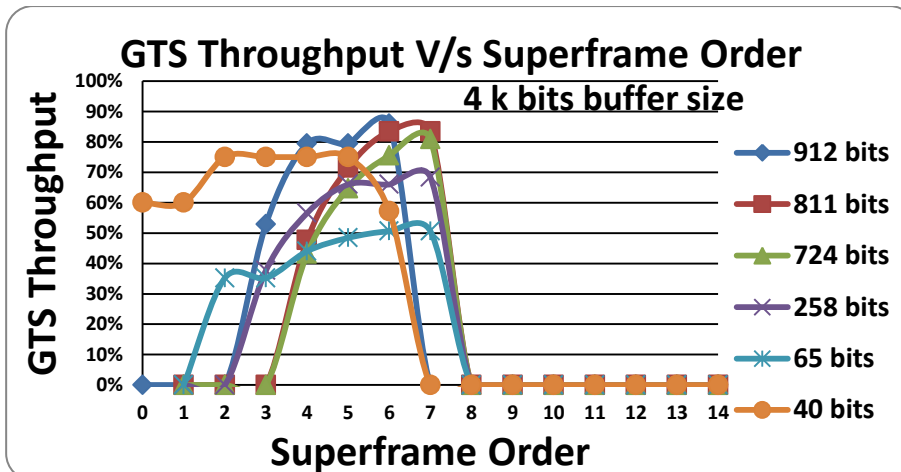


Figure 6.5(b): GTS throughput v/s superframe order for 4 k buffer capacity

¹ Published a paper, Ms. Sonal J. Rane "THROUGHPUT OPTIMIZATION IN IEEE 802.15.4 USING GTS MECHANISM", International Journal of Latest Research in Science and Technology, Volume 2, Issue 1: Page No.544-547, January-February (2013), ISSN(Online): 2278-5299

In these graphs, the corresponding packet size has reached its saturation throughput when the curve reaches its maximum peak value and plateaus. Throughput gradually decreases after reaching certain SO. The higher the SO, the larger the SD is. This creates unproductive service in time slot window duration with the corresponding buffer size and packet generation. This can be observed for SO beyond 7. Also higher SO values are not suitable for WSN application as Higher SO values, SD will be higher. This will provide more time duration to transmit the data but simultaneously delay will increase and throughput will decrease. At the lower SO, the duration of the superframe is very small so that only a packet size of 40 bits can conform.

6.7.2 Impact of Duty cycle on global statistics throughput.

Now consider Global statistics throughput, with and without inactive period. When only active period (100 % duty cycle) is considered, more throughput is achieved compared to considering inactive period ($BO=SO+1$) (Figure 6.6).

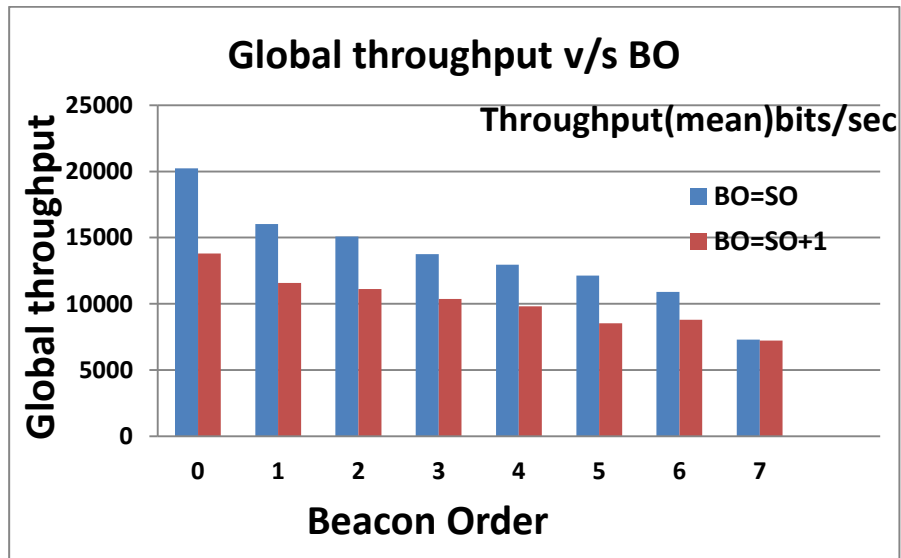


Figure 6.6: Global Throughput v/s BO

As the inactive part increases, the throughput drops significantly. This is because as we increase inactive period, that may save power consumption but data transmission does not take place, which will decrease the global throughput.

For the high SO values we get less GTS throughput and higher than 7 values, we don't get any throughput. This is because as SO values of the superframe increases, the active period increases (for 100% duty cycle) which also increases transmitted packet, but simultaneously delay will increase and overall global throughput will decrease.

6.7.3 Impact of Buffer size on Packet Medium Access Delay

This section evaluates and compares the Packet Medium Access Delay during one time slot of GTS when inter-arrival time is set to 0.001824 s for different values of the SO and for different packet and buffer sizes.

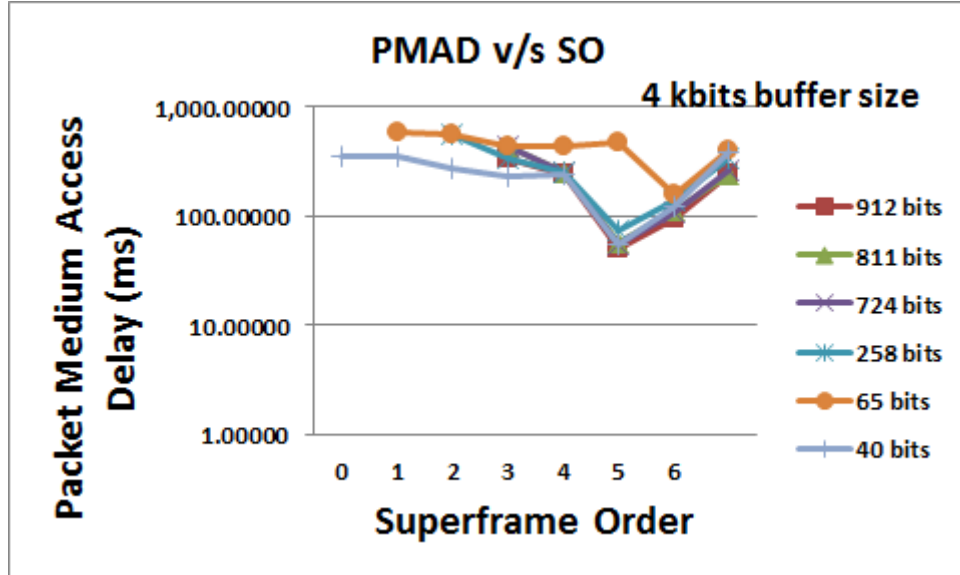


Figure 6.7 (a): PMAD v/s superframe order for 4 k buffer capacity

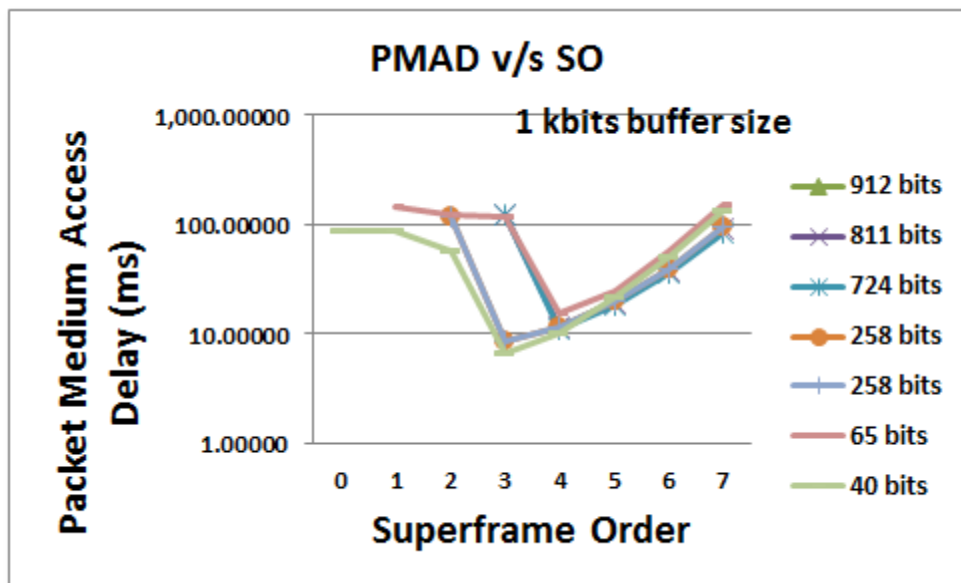


Figure 6.7 (b): PMAD v/s superframe order for 1 k buffer capacity

When SO increases, 0.001824 s inter-arrival time records a bigger range of Packet Medium Access Delay (PMAD) for both the cases (1kbits and 4kbits). This indicates the significant relationship of inter-arrival time, packet size and buffer size. As shown in Figure 6.7(a) and (b) initially PMAD is somewhat stable and then starts to decrease

slightly. The point at which the PMAD decreases slightly indicates that the PMAD contribution originates from the long SD produced by the SO. The PMAD measurement is similar for the same SO regardless of the buffer size until it reaches that slightly decreased point. Also for higher SO values, all frames stored in the buffer and transmitted during one GTS and the delay grows with SO. For higher values of SO, delay will increase.

Readings for the PMAD for SO values at which SO is less than that decreased point, is not the same for each buffer size compared to the same scenario for inter-arrival of 0.001824s. Thus, with a high arrival rate, buffer size plays an important role towards packet end-to-end delay. For real-time traffic, the end-to-end delay is to be limited to a maximum of 250 ms.

WSN applications present real-time requirements such as bounded PMAD and high QoS sensitivity[16]. For instance, in voice communication, interactive voice requires PMAD latency of 250 ms or less, beyond which users notice a drop in interaction quality [19]. Similarly, video streaming over WSN requires a BER lower than 10^{-4} [20]. Inline with this, the IEEE 802.15.4 standard [1] provides GTS for resource reservation for those applications that have bandwidth requirements or low delay constraints.

6.7.4 Impact of the Superframe Orders on Wasted Bandwidth

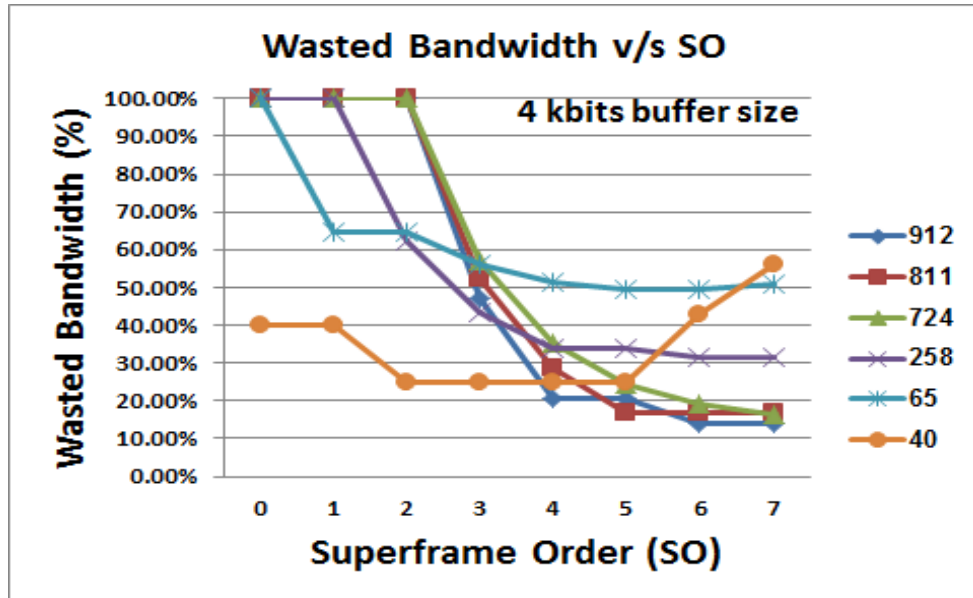


Figure 6.8: Wasted Bandwidth v/s superframe order for 4 k buffer capacity

Since the frames are transmitted without acknowledgement, the wasted bandwidth can only result from IFS or waiting for a new frame if the buffer is empty, as depicted in

Figure 6.3. Observe in Figure 6.8 that the wasted bandwidth is high for low superframe orders. This is due to the impact of IFS, since the time slot durations are too small to send high amount of data.

Also, the devices request for GTS allocation through the GTS request command by specifying the number of slots needed and direction of GTS transmission (from or to the coordinator). The GTS slots are allocated in every superframe so they consume a significant bandwidth of the superframe duration. Therefore, inefficient allocation of GTS can lead to significant loss of bandwidth and degradation of the overall system performance [7].

As discussed in chapter 5, Superframe structure is divided in 16 slots and PAN coordinator allocates the GTS slots to the standard slot duration i.e., $SD/16$ in superframe. Because of a small fraction of the allocated slot is used for data transmission, the remaining bandwidth is wasted in every GTS slot in every superframe for the application which consists low average packet arrival rate and low data payload. So for this, a proposed method having a modified superframe structure is used which main intense is to save the wasted bandwidth so that it will optimize the throughput.

In standard IEEE 802.15.4, T_s is defined as

$$T_s = SD/16.$$

While in new proposed method slot duration is considered half of the standard value,

$$T_s = SD/32.$$

Now one GTS allocation means it contains half time duration to transmit the data. This method can be applicable for the application which consist low average packet arrival rate and low data payload.

The proposed method is carried out in Matlab by modifying the superframe duration. The following Table 6.4 consists of the simulation parameters varied in simulation and measured parameter (Wasted Bandwidth), considering standard and modified structure.

SO	BO	Superframe Duration(SD)	Beacon Interval(BI)	Duty Cycle (%)	Time Slot Duration (TS)		Wasted Bandwidth	
					For Standard IEEE 802.15.4	For Modified Proposed Method	For Standard IEEE 802.15.4	For Modified Proposed Method
1	1	30.72	30.72	100	1.92	0.96	10.4167	4.6875
2	2	61.44	61.44	100	3.84	1.92	13.0208	5.20833
3	3	122.88	122.88	100	7.68	3.84	13.0208	6.51042
4	4	245.76	245.76	100	15.36	7.68	13.0208	6.51042
5	5	491.52	491.52	100	30.72	15.36	13.3464	6.51042
6	6	983.04	983.04	100	61.44	30.72	13.4603	6.67318
7	7	1966.08	1966.08	100	122.88	61.44	13.4847	6.73014
8	8	3932.16	3932.16	100	245.76	122.88	13.4725	6.74235

Table 6.4 Simulation Parameters and resultant parameter

Figure 6.9 shows the graphs comparing wasted bandwidth for different SO values in standard IEEE 802.15.4 and modified proposed method.

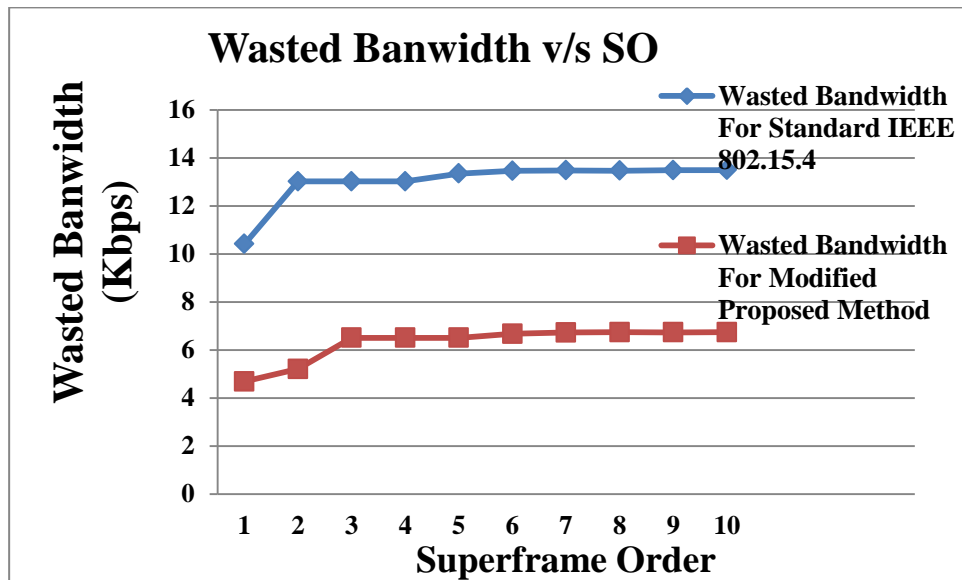


Figure 6.9: Comparisons of Wasted Bandwidth Results

From the Figure 6.9 it can be seen that in proposed method wasted bandwidth decreases compare to the standard IEEE 802.15.4. It is already discussed that the GTS allocation mechanism presents a worst behavior in terms of maximum throughput for

different superframe orders due to a high amount of wasted bandwidth in large GTSs. Therefore, as wasted bandwidth decreases we get optimum throughput for a given GTS allocation. This can be used in the applications to wide variety of WPAN applications such as home automation, remote sensing where the devices in the network have a low average packet arrival rate, small payload and requires minimal dedicated bandwidth for contention free transmissions.

Summary:

In this chapter GTS (local) throughput and Global throughput for the WSN beacon enabled mode are evaluated for GTS Mechanism. The evaluation is performed using IEEE 802.15.4 OPNET simulation model. The maximum utilization of the allocated GTS is achieved for superframe orders $SO = [2, 3, 4, \text{ and } 5]$. As inactive period increases, throughput decreases which is influenced by processing, transmitting, propagation, and queuing delays. The throughput can be increased by decreasing wasted bandwidth which occurs because of a fraction of the slot can be used during transmission of data. This can be resolved by splitting the slots which make superframe structure with 32 slots. This method can be applicable only for small data transmission which can accommodate with this new slot duration. If data is large which cannot accommodate with this modified slot duration then modified method may not give optimum solution.



Chapter 7



Soft Computing based WSN



This chapter briefs about the Soft Computing Techniques such as Artificial Neural Network (ANN), Adaptive Neuro Fuzzy Inference System (ANFIS). Performance of WSN is evaluated based on these Soft Computing Techniques. Optimization using ANN and ANFIS decide the Packet size of data transmitted by WSN Nodes.

7.1 Soft Computing Models

Soft computing is an emerging approach to computing which parallels the remarkable ability of the human mind to reason and learn in an environment of uncertainty and imprecision [1]. It is a parasol term for a collection of computing techniques. Soft Computing (SC), a term originally coined by Zadeh (1994) is a sub-field of Artificial Intelligence. In his words: “Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty and partial truth. In effect, the role model for soft computing is the human mind. The guiding principle of soft computing is: Exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness and low solution cost”. The main techniques in soft computing are evolutionary computing, artificial neural networks and fuzzy logic. Each technique can be used separately, but a powerful advantage of soft computing is the complementary nature of the techniques. Used together they can produce solutions to problems that are too complex or inherently noisy to tackle with conventional mathematical methods. Soft computing can be used to address a very wide range of problems in all industries and business sectors. In general Soft Computing is a good option for complex systems [2], where:

- ✖ The system is non-linear, time-variant or ill defined.
- ✖ The variables are continuous.
- ✖ A mathematical model is either too difficult to encode, does not exist or is too complicated and expensive to be evaluated.
- ✖ There are noisy or numerous inputs.
- ✖ An expert is available who can outline the rules-of-thumb that should determine the system behaviour.

7.2 Artificial neural networks

One type of network sees the nodes as ‘artificial neurons’. These are called artificial neural networks (ANNs). An artificial neuron is a computational model inspired in the natural neurons. Natural neurons receive signals through *synapses* located on the

dendrites or membrane of the neuron. When the signals received are strong enough (surpass a certain *threshold*), the neuron is *activated* and emits a signal through the *axon*. This signal might be sent to another synapse, and might activate other neurons. The complexity of real neurons is highly abstracted when modelling artificial neurons. These basically consist of *inputs* (like synapses), which are multiplied by *weights* (strength of the respective signals), and then computed by a mathematical function which determines the *activation* of the neuron. Another function (which may be the identity) computes the *output* of the artificial neuron (sometimes in dependence of a certain *threshold*). ANNs combine artificial neurons (as shown in Figure 7.1) in order to process information [3].

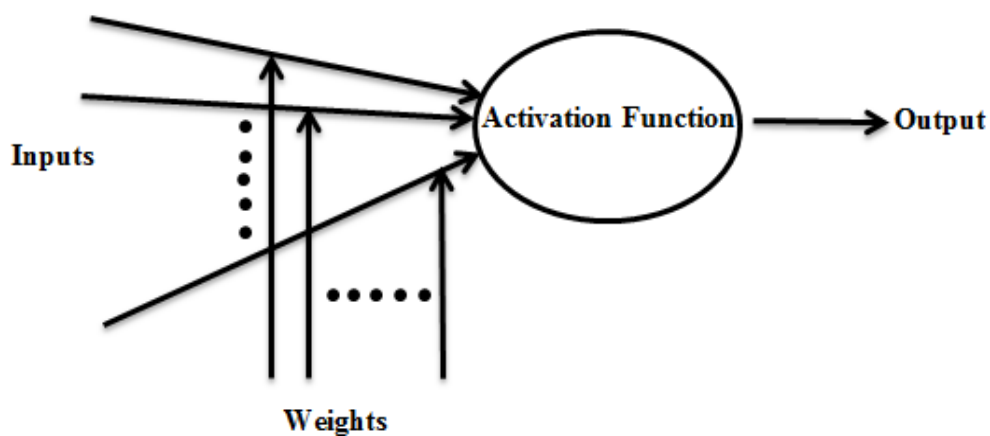


Figure 7.1: an Artificial Neuron

There are different ways of defining what the ANN are, from short and generic definitions to the ones that try to explain in a detailed way what means a neural network or neural computing. For this situation, the definition that was proposed by Teuvo Kohonen appears below:

“Artificial Neural Networks are massively interconnected networks in parallel of simple elements (usually adaptable), with hierarchic organization, which try to interact with the objects of the real world in the same way that the biological nervous system does.”

Basic building block of every artificial neural network is artificial neuron, that is, a simple mathematical model (function) [4, 5, 6]. Such a model has three simple sets of rules: multiplication, summation and activation. At the entrance of artificial neuron the inputs are weighted what means that every input value is multiplied with individual weight. In the middle section of artificial neuron is sum function that sums all weighted inputs and bias. At the exit of artificial neuron the sum of previously weighted inputs and bias is passing through activation function that is also called transfer function which defines the properties of artificial neuron and can be any mathematical function. We

choose it on the basis of problem that artificial neuron (artificial neural network) needs to solve and in most cases we choose it from the set of functions: *Step function*, *Linear function* and *Non-linear (Sigmoid) function*.

7.2.1 Basics of Artificial Neural Networks (ANN)

ANN consists of two or more number of artificial neurons. ANN solves complex real-life problems by processing information in a non-linear, distributed, parallel and local way. The way that individual artificial neurons are interconnected is called topology under layers viz., input, hidden, output which represents architecture or graph of an ANN.

7.2.2 Types of ANN Learning

There are three major learning paradigms; supervised learning, unsupervised learning and reinforcement learning. Usually they can be employed by any given type of artificial neural network architecture. Each learning paradigm has many training algorithms.

✂ Supervised learning

Supervised learning is a machine learning technique that sets parameters of an artificial neural network from training data. Supervised learning is a process of training a neural network by giving it examples of the task we want it to learn. i.e., it is learning with a teacher. The way this is done is by providing a set of pairs of vectors (patterns), where the first pattern of each pair is an example of an input pattern that the network might have to process and the second pattern is the output pattern that the network should produce for that input which is known as a target output pattern for whatever input pattern.

Thus, supervised learning means “a learning process in which, change in a network's weights and biases are due to the intervention of any external teacher. The teacher typically provides output targets.”

✂ Unsupervised learning

Unsupervised learning is a machine learning technique that sets parameters of an artificial neural network based on given data and a cost function which is to be minimized. Cost function can be any function and it is determined by the task formulation. Unsupervised learning is mostly used in applications that fall within the domain of estimation problems such as statistical modelling, compression, filtering, blind source separation and clustering. In unsupervised learning we seek to determine how the

data is organized. It differs from supervised learning and reinforcement learning in that the artificial neural network is given only unlabelled examples.

✂ Reinforcement learning

Sometimes if it is not require computing exact error between the desired and the actual network response, and for each training example the network is given a pass/fail signal by the teacher, then it is called Reinforcement learning which is a special type of supervised learning. If a fail is assigned, the network continues to readjust its parameters until it achieves a pass or continues for a predetermined number of tries, whichever comes first.

7.2.3 Back Propagation Network (BPN)

Back propagation is a systematic method for training multi-layer artificial networks [7]. It has a mathematical foundation that is strong if not highly practical. It is a multi-layer forward network using extend gradient descent based delta learning rule, commonly known as back propagation (of errors) rule.

Back propagation provides a computationally efficient method for changing the weights in a feed forward network, with differentiable activation function units, to learn training a set of input-output examples. Being a gradient descent method it minimizes the total error of the output computed by the net. The network is trained by supervised learning method. The aim of this network is to be train the net to achieve a balance between the ability to respond correctly to the input patterns that are used for training and the ability to provide good response to the input that are similar. The training algorithm of back propagation involves four stages, viz.

1. Initialize of weights
2. Feed forward
3. Back propagation
4. Updating of the weights and biases.

7.3 Adaptive Neuro Fuzzy Inference System (ANFIS)

The acronym ANFIS derives its name from adaptive neuro-fuzzy inference system. Using a given input/output data set, the toolbox function `anfis` constructs a fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using either a back-propagation algorithm alone or in combination with a least squares type of method.

ANFIS is a network-type structure similar to that of a neural network, which maps inputs through input membership functions and associated parameters, and then through output membership functions and associated parameters to outputs, can be used to interpret the input/output map. The parameters associated with the membership functions will change through the learning process. The computation of these parameters (or their adjustment) is facilitated by a gradient vector, which provides a measure of how well the fuzzy inference system is modelling the input/output data for a given set of parameters.

7.4 Matlab Development tools

The development tools such as MATLAB, SIMULINK and various tools boxes are covered in the following section.

7.4.1 MATLAB

MATLAB [9] is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and FORTRAN.

- ✂ **Desktop Tools and Development Environment:** - This is the set of tools and facilities that help to use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path.
- ✂ **The MATLAB Mathematical Function Library:** - This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic.
- ✂ **The MATLAB Language:** - This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.
- ✂ **External Interfaces:** - The external interfaces library allows to write c/c++ and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), for calling MATLAB as a computational engine, and for reading and writing MAT-files.
- ✂ **Graphics:** - MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image

processing, animation, and presentation graphics. It also includes low-level functions that allow to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

The main MATLAB window is given in Figure 7.2

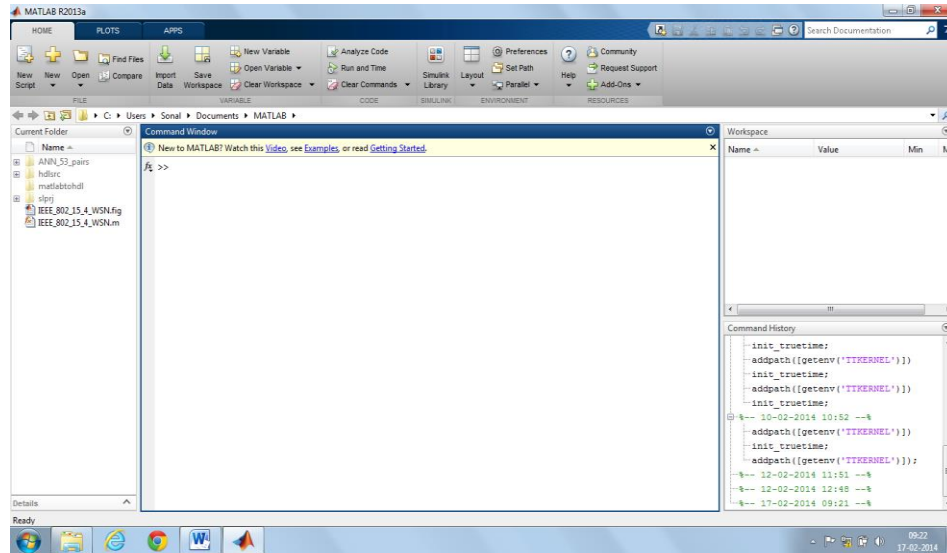


Figure 7.2: MATLAB Command Window

7.4.2 Simulink

Simulink [9] is an environment for multidomain simulation and Model-Based Design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that let you design, simulate, implement, and test a variety of time-varying systems, including communications, controls, signal processing, video processing, and image processing.

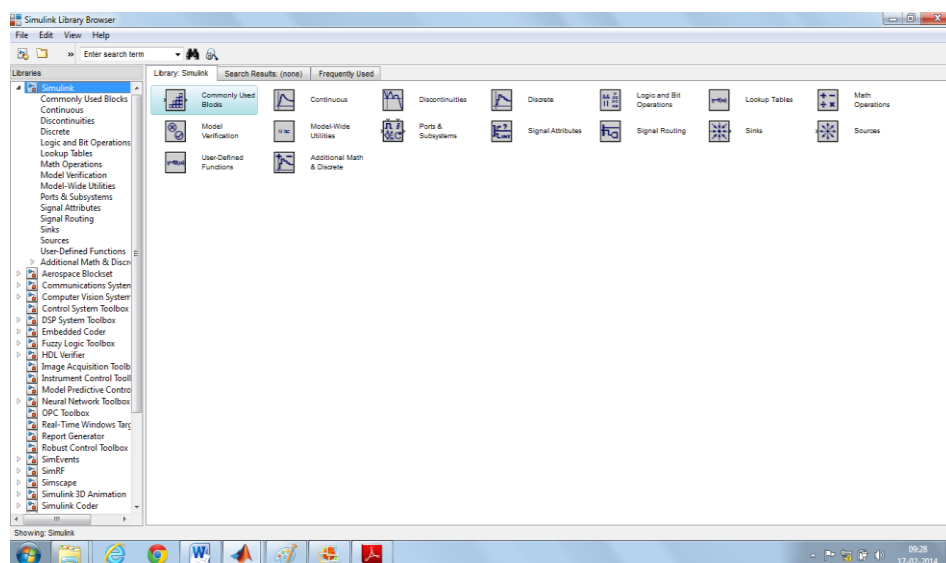


Figure 7.3: Simulink View

The default Simulink view is given in Figure 7.3.

7.4.3 Toolboxes

✂ Neural network toolbox

There are four different levels at which the Neural Network Toolbox software can be used.

- ❖ The first level is represented by the Graphical User Interfaces. These provide a quick way to access the power of the toolbox for many problems of function fitting, pattern recognition, clustering and time series analysis.
- ❖ The second level of toolbox use is through basic command-line operations. The command-line functions use simple argument lists with intelligent default settings for function parameters.
- ❖ A third level of toolbox use is customization of the toolbox. This advanced capability allows you to create your own custom neural networks, while still having access to the full functionality of the toolbox.
- ❖ The fourth level of toolbox usage is the ability to modify any of the M-files contained in the toolbox. Every computational component is written in MATLAB code and is fully accessible.

✂ ANFIS Editor

To start ANFIS editor [10] GUI, on the MATLAB command prompt type the following.

```
>>anfisedit
```

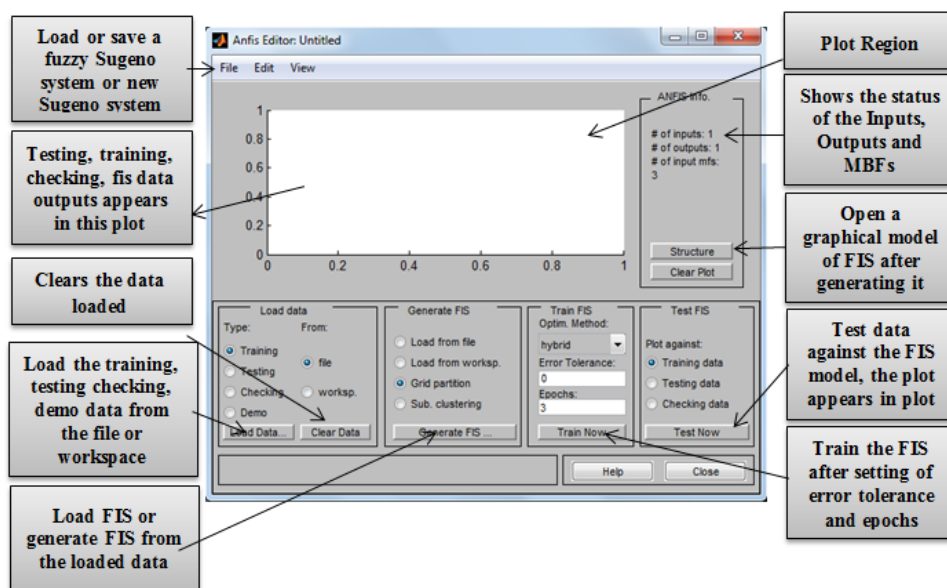


Figure 7.4: ANFIS Editor GUI

The ANFIS Editor GUI is shown in Figure 7.4. The main functions in the anfisedit are the loading of the training data from the work space. Then after by loading the training data into the anfiseditor select the sub clustering button to generate the FIS model. After doing these steps now select the training algorithm and error tolerance and number of epochs. Now press the train button which will train the Neuro-fuzzy controller and it generates the training rules itself.

✂ GENERAL DESIGN METHODOLOGY

- 1) Analyze the problem and find whether it has sufficient elements for a neural network solution. Consider alternative approaches. A simple DSP/ASIC based direct solution may be satisfactory.
- 2) If the ANN is to represent a static function, then a three-layer feed forward network should be sufficient. For a dynamic function select either a recurrent network or a time delayed network. Information about the structure and order of the dynamic system is required.
- 3) Select input nodes equal to the number of input signals and output nodes equal to the number of output signals and a bias source. For a feed forward network, select the initially hidden layer neurons typically mean of input and output nodes.
- 4) Create an input/output training data table. Capture the data from an experimental plant or simulation results, if possible.
- 5) Select input scale factor to normalize the input signals and the corresponding output scale factor for denormalization.
- 6) Select generally a sigmoidal transfer function for unipolar output and a hyperbolic tan function for bipolar output.
- 7) Select a development system, such as s Neural Network Toolbox in MATLAB.
- 8) Select appropriate learning coefficients and momentum factor.
- 9) Select an acceptable training error and a number of epochs. The training will stop whichever criterion was met earlier.
- 10) After the training is complete with all the patterns, test the network performance with some intermediate data points.
- 11) Finally, download the weights and implement the network by hardware or software.

7.5 ANN based GTS Mechanism

Analysis of GTS mechanism (discussed in chapter 6) is evaluated for enhanced GTS throughput using Artificial Neural Network (ANN) [8, 11] soft computing technique in OPNET Modeler, which can be achieved due to the packet size based on data rate and Interarrival time.

Here two inputs are considered to ANN for optimizing the GTS throughput as shown in Figure 7.5. A packet interarrival time and data rate input decides the best packet size output through trained ANN. Since it is possible to generate I/O training pairs from the existing environment and classical procedures, a feed forward ANN with multi layers Perception model, with 2 nodes in the input Layer, 10 nodes in hidden layer and 1 node in the output layer is selected. The input layer, hidden and output layers have a sigmoid tan-type activation function to produce outputs.

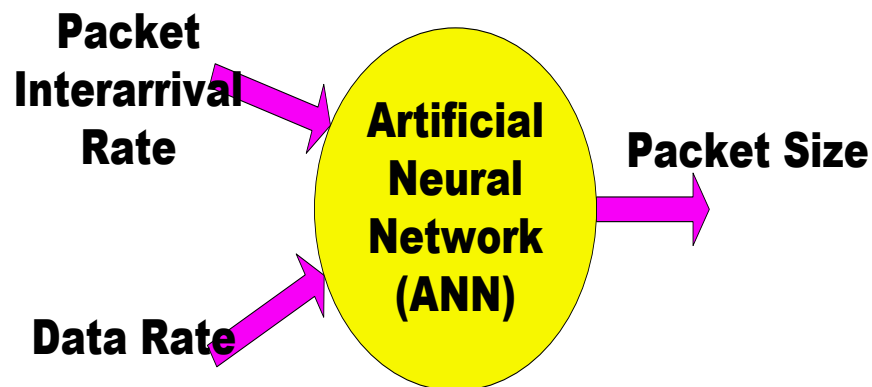


Figure 7.5: ANN with respective inputs and outputs

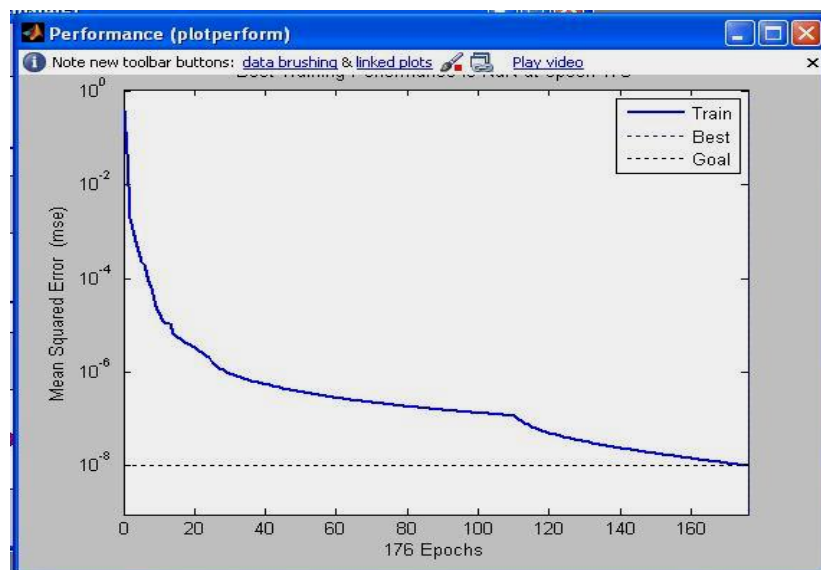


Figure 7.6 Training of Artificial Neural Network for Packet size

Training of neural network tries to achieve the goal of $1E-5$ within 20 epochs; the goal performance is 9.95×10^{-5} . The Figure 7.6 shows the training of the neural network.

7.5.1 Implementation of ANN GTS MECHANISM^{1,2}

The simulation results were carried out using OPNET simulator and MATLAB to evaluate the performance of the GTS Mechanism where Simulation model has one pan coordinator and one end device (GTS enabled). This configuration is discussed in chapter 6.

The impact of data Rate, Packet Size, Packet Interarrival time, Buffer Size is evaluated on GTS throughput using traditional and soft computing method for different values of SO (=BO) [12]. The intense of this section is to optimize the GTS throughput during one time slot of GTS when inter arrival time is set to 0.001824 and for different values of the SO [13]. Since the frames are transmitted without acknowledgements, the underutilized bandwidth can only result from IFS or from intermittent data arrival at the buffer.

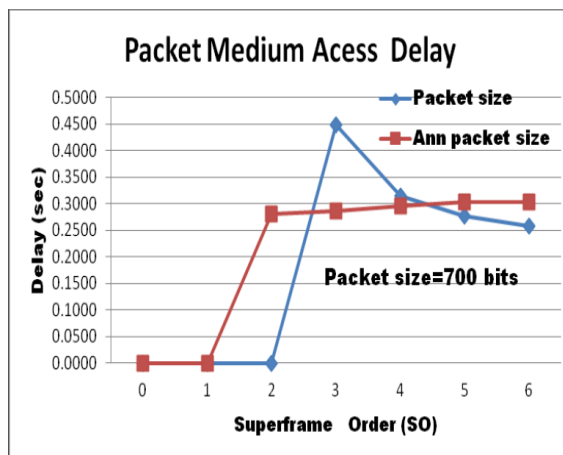


Figure 7.7(a): Packet Medium Access Delay v/s Superframe Order for 700 bits Packet Size

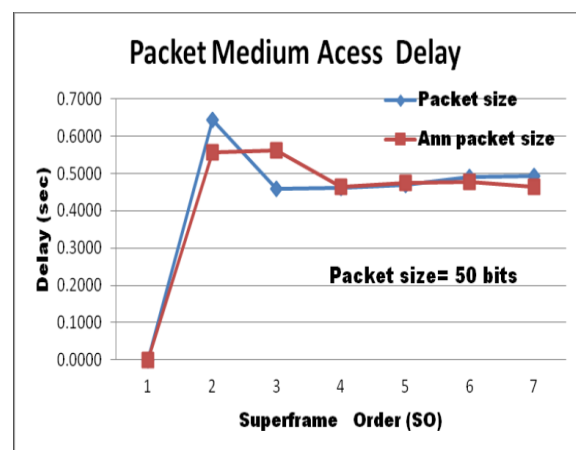


Figure 7.7(b): Packet Medium Access Delay v/s Superframe Order for 50 bits Packet Size

Figure 7.7(a) and (b) plot the packet medium access delay (sec). The lowest delay is achieved for SO values equal to 2, 3. For SO values higher or equal to 5, all frames

¹ Present a paper: Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma, "Soft Computing Technique Based Throughput Optimization of GTS mechanism for IEEE 802.15.4 Standard" in International Conference on Soft Computing and Software Engineering 2013 (SCSE'13), in San Francisco, California, USA.

² Published a paper: Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma, "Soft Computing Technique Based Throughput Optimization of GTS mechanism for IEEE 802.15.4 Standard" in International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, pp. 668-673, 2013, Doi: 10.7321/jscse.v3.n3.102

stored in the buffer and transmitted during one GTS and the delay grows with SO. For higher values of SO, delay will increase.

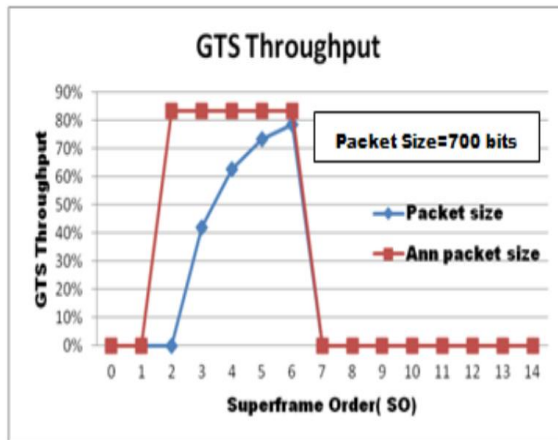


Figure 7.8(a): GTS Throughput v/s Superframe Order for 700 bits Packet Size

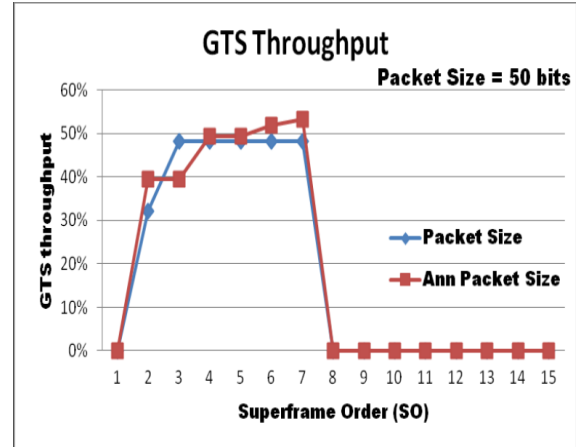


Figure 7.8(b): GTS Throughput v/s Superframe Order for 50 bits Packet Size

Figures 7.8(a) and (b) plot the GTS throughput for traditional GTS mechanism and Soft GTS mechanism. Figure 7.9 (a) and (b) show the Wasted Bandwidth for Traditional and Soft GTS mechanism.

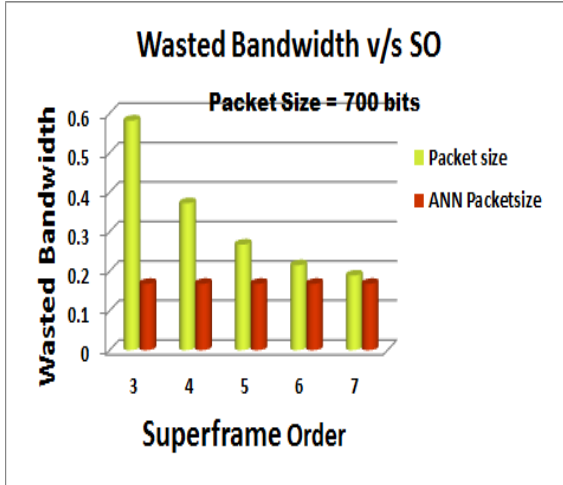


Figure 7.9(a): Wasted B.W. v/s Superframe Order for 700 bits Packet Size

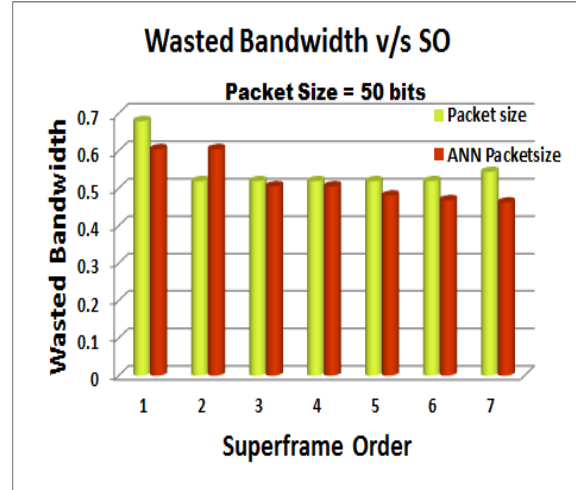


Figure 7.9(b): Wasted B.W. v/s Superframe Order for 50 bits Packet Size

It can observe that when Soft GTS mechanism is applied, GTS throughput is increased, Wasted Bandwidth and PMAD decrease for different SO values compare to Traditional Simulation method. That means considered parameters are optimized after applying the soft computing method to predict the packet size.

7.6 ANFIS based GTS Mechanism

ANFIS is a network-type structure similar to that of a neural network, which maps inputs through input membership functions and associated parameters, and then through output membership functions and associated parameters to outputs, can be used to interpret the input/output map. The parameters associated with the membership functions will change through the learning process. The computation of these parameters (or their adjustment) is facilitated by a gradient vector, which provides a measure of how well the fuzzy inference system is modelling the input/output data for a given set of parameters [13].

Once the gradient vector is obtained, any of several optimization routines could be applied in order to adjust the parameters so as to reduce some error measure (usually defined by the sum of the squared difference between actual and desired outputs). ANFIS uses either back propagation or a combination of least squares estimation and gradient method for membership function parameter estimation. The GTS Mechanism is also optimized using ANFIS [14] with same inputs. The implementation of ANFIS is accomplished using ANFIS editor available in MATLAB. The Sugeno type of inference system is used.

A packet Interarrival time and data rate input decides the best packet size output through trained ANFIS. ANFIS is used offline with OPNET modeler for IEEE 802.15.4 GTS mechanism and evaluated the performance. Total (53x53) 2809 pairs are used to minimize the error between trained output and desired output as shown in Figure 7.10.

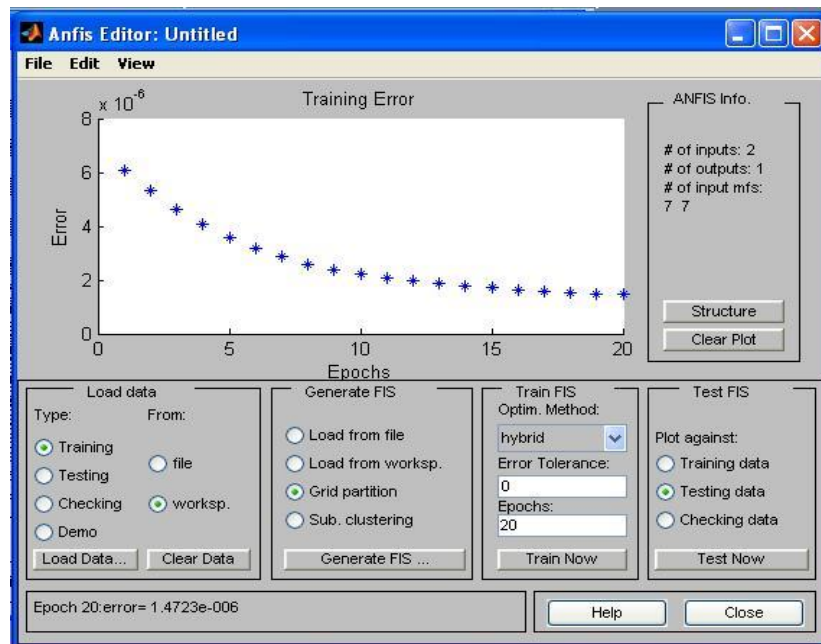


Figure 7.10: ANFIS training

ANFIS architecture was trained with Number of nodes are 131 with training data pairs of 2809 using 49 fuzzy rules.

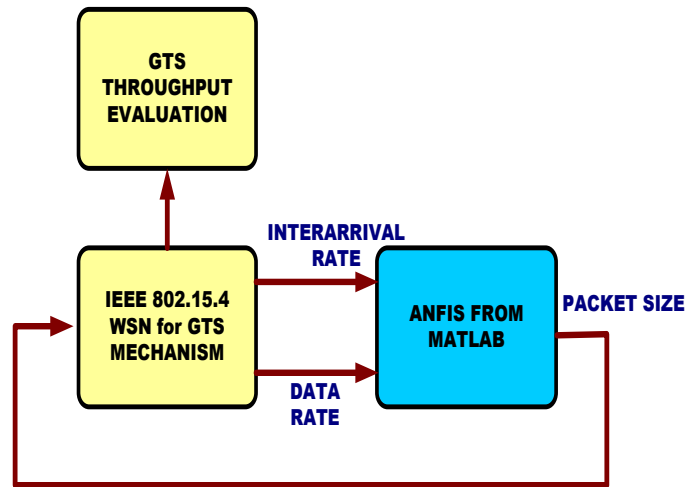


Figure 7.11: System Setup for ANFIS based GTS Mechanism

The system setup for evaluation of performance of GTS mechanism for WSN using ANFIS in OPNET and MATLAB is shown in Figure 7.11.

7.7 Comparison of Traditional method, ANN, ANFIS

Figure 7.12 shows the results analysis among traditional, ANN and ANFIS.



Figure 7.12(a): Result analyses among traditional, ANN and ANFIS GTS Mechanism

Figure 7.12(b): Result analyses among traditional, ANN and ANFIS GTS Mechanism

From Figure 7.12(a) and (b) it can observe that by applying ANFIS soft computing technique better performance (high throughput and low Wasted Bandwidth) is achieved for packet sizes decided by ANFIS.

7.8 Soft GTS Mechanism Simulator³

MATLAB Graphical User Interface [15] development environment provides a set of tools for creating graphical user interfaces (GUIs). These tools simplify the process of laying out and programming GUIs.

Using the GUIDE Layout Editor, you can populate a GUI by clicking and dragging GUI components—such as axes, panels, buttons, text fields, sliders etc.—into the layout area. You can also create menus and context menus for the GUI. From the Layout Editor, you can size the GUI, modify component look and feel, align components, set tab order, view a hierarchical list of the component objects, and set GUI options.

GUIDE automatically generates an M-file that controls how the GUI operates. This M-file provides code to initialize the GUI and contains a framework for the GUI callbacks—the routines that execute when a user interacts with a GUI component. Using the M-file editor, you can add code to the callbacks to perform the functions you want.

Figure 7.13 shows the snap shot of Graphical User Interface SOFT GTS MECHANISM SIMULATOR, designed to do the performance analysis of GTS Mechanism using IEEE 802.15.4 OPNET Simulation model and Adaptive soft computing method having following facilities [16]:

- To change the IEEE 802.15.4 Standard parameters i.e., SO (Superframe duration) and BO (Beacon Order).
- To change the different nodes considered in input, hidden and output layers in soft computing technique.
- To view the different structures, literatures which are taken to carry out this research work.
- To view results of different statistics i.e., GTS throughput, wasted bandwidth by varying different parameters.

3

Published a paper: Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma,” Soft GTS Mechanism Simulator in IEEE 802.15.4 WSN” in International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X ,Volume 3, Issue 9, September 2013.

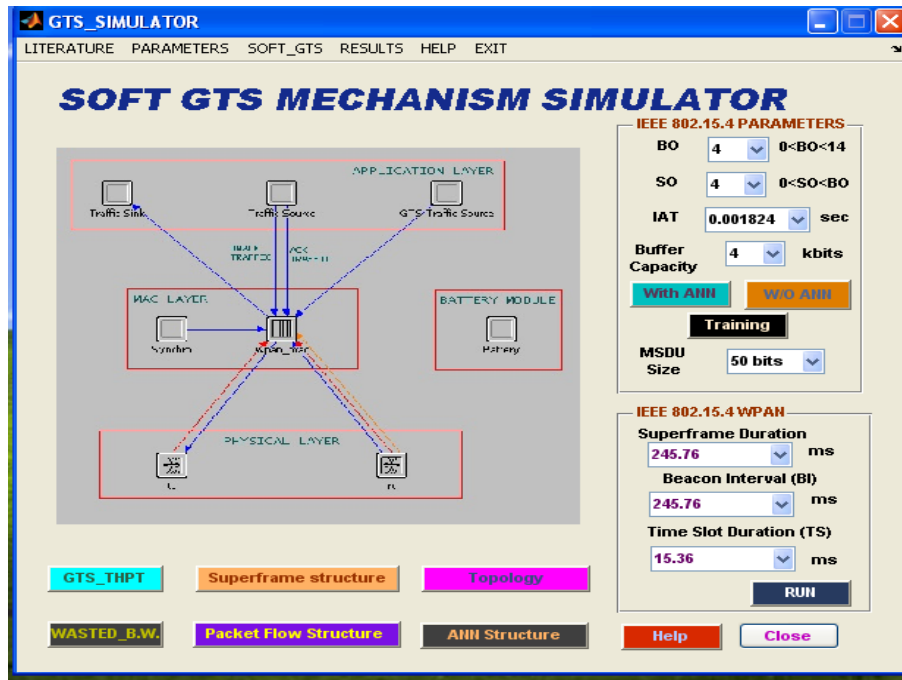


Figure 7.13: Soft GTS Mechanism Simulator

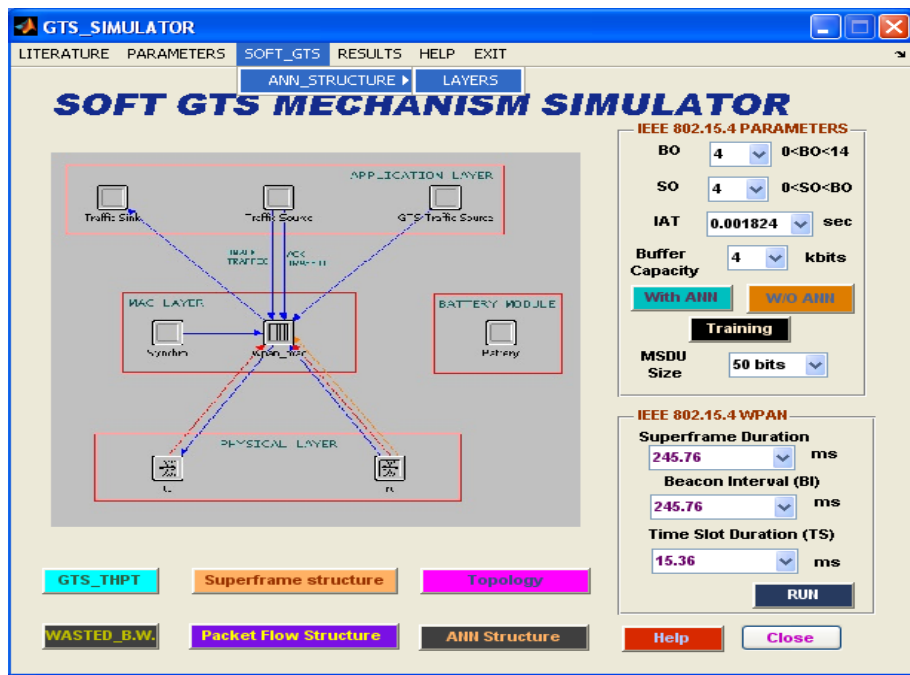


Figure 7.14: Snapshot of menus facility provided in given simulator

It has different menus in menu bar (Refer Figure 7.14), such as Literature, selecting parameters, changing layers in proposed soft computing method, results, help and exit. Literature menu provides different references, pdf files, based on which this research is carried out. SOFT GTS menu provides facility in which input, hidden and output nodes of ANN can be varied to get best output. Result menu gives the graphical result of different parameters that we have chosen. It also provides the online help as one

can require anytime and the exit facility to close the simulator only or to close the whole matlab. In GUI simulator, by clicking the push button **GTS_THPT** and **WASTED_B.W**, graphs are displayed in the display section for different values of SO and BO values. The snapshot is shown in below Figure 7.15. The SO and BO values can be varied for a given condition by clicking the push button SO and BO, designed in right upper panel of IEEE 802.15.4 Parameters.

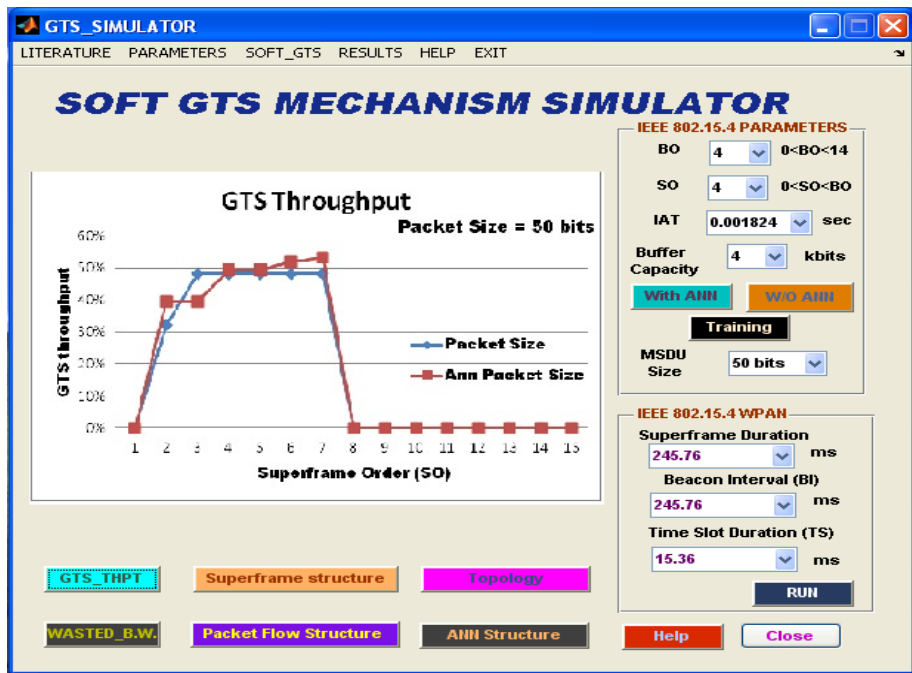


Figure 7.15: Snapshot of GTS Throughput result in designed simulator

This GUI simulator also provides the result with and without applying ANN technique. When ANN push button is selected, the training graph is displayed by clicking the Training button. In below side of GUI, Push button for superframe structure, Packet Flow Structure, ANN structure and considered topology structure are given. In upper panel-IEEE 802.15.4 Parameters (Right Side), values of parameters SO and BO can be selected within the range specified. The corresponding SI and BI values are set, in below panel-IEEE 802.15.4 WPAN shown in Figure 7.15.

Summary:

The theoretical background of soft computing techniques such as ANN and ANFIS is summarized and described. Toolboxes available for deploying soft computing techniques in MATLAB and used in our research work for the design and testing of proposed techniques are described in detail. Procedural steps to be followed in each trait are discussed in detail. Also GTS Mechanism can be optimized by deciding packet size using soft computing Techniques such as ANN and ANFIS.



Chapter 8



Embedded Hardware : WSN



This chapter divides in two sections A and B. A section implements trained Artificial Neural Network (ANN) in MATLAB using VHDL programming language and then realized on FPGA kit. B section describes the real time test bed hardware implementation of Wireless Sensor Network (WSN).

SECTION A: Configuration of ANN on FPGA Kit

8.1 Short introduction to FPGAs

Field Programmable Gate Arrays (FPGAs) made their appearance in 1985 when Xilinx started to manufacture the XC2064 [1]. The general architecture of an FPGA structure is composed of four basic reconfigurable elements: Programmable Logic Blocks (PLBs) which is the most significant part that provides physical support for the program downloaded on FPGA, embedded memory, programmable I/O cells which provides input and output for FPGA and makes it possible to communicate outside the FPGA, and programmable interconnections (Programmable Interconnect (PI)) which connects the different part of FPGA and allows them to communicate with each other [2,3]. The way in which these elements are distributed inside the device defines the technical characteristics of each FPGA family. These structures consist of routing channels and programmable switches. Routing process is effectively connection logic blocks exist different distance the others [4].

FPGAs can be programmed via interfaces based on Hardware Description Languages (HDL); the most popular one is the Very High Speed Integrated Circuit (VHSIC) Hardware Description Language, commonly known as VHDL. The process to design an application with an FPGA consist of six main phases: 1) definition of the initial requirements; 2) choice of the appropriate device; 3) writing of the VHDL code; 4) synthesis to map the application onto the resources of the FPGA; 5) simulation; 6) programming of the FPGA (if the simulation succeeds).

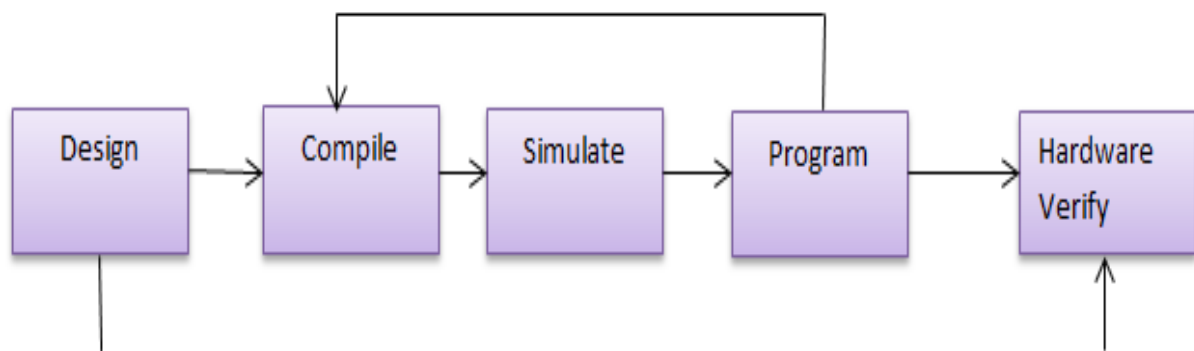


Figure 8.1: Design Flow

The flow in the FPGA hardware is shown in Figure 8.1.

8.2 FPGA design implementation of ANN

FPGAs are chosen for implementation ANNs with the following reason:

- ✖ They can be applied a wide range of logic gates starting with tens of thousands up to few millions gates.
- ✖ They can be reconfigured to change logic function while resident in the system.
- ✖ FPGAs have short design cycle that leads to fairly inexpensive logic design.
- ✖ FPGAs have parallelism in their nature. Thus, they have parallel computing environment and allows logic cycle design to work parallel.
- ✖ They have powerful design, programming and syntheses tools.

Artificial neural network based on FPGAs has fairly achieved with classification application. The programmability of reconfigurable FPGAs yields the availability of fast special purpose hardware for wide applications.

There are two problems during the hardware implementation of ANNs. How to balance between the need of reasonable precision (number of bit), that is important for ANN and the cost of more logic area associated with increased precision. How to choose a suitable number format that dynamic range is large enough to guarantee that saturation will not occur for a general-purpose application. So before beginning ANN's based FPGAs system design with VHDL, number format (floating point, fixed point etc.) and precision which used for inputs, weighs and activation function must be considered. This important that precision of the numbers must be as high as possible, are used during training phase. This is because precision has a great impact in the learning phase [5]. However, low precision is used during the propagation phase [6]. So especially in classification's applications the resulting errors will be small enough to be neglected [6, 7, 8]. Floating point offers the greatest amount of dynamic range, making it suitable for any application so it would be the ideal number format to use. In this implementation we have used the fractional fixed-point representation to represent the real numbers.

A 2-10-1 feed forward network (two neurons in the input layer, ten neurons in the hidden layer and one neuron in the output layer) is implemented on a Xilinx Spartan-6 LX45 FPGA demo board (features are covered in appendix)[9]. The design implementation is shown in Figure 8.2. It shows the total design flow using MATLAB and Xilinx. The MATLAB program consists of the built and learning programs of ANN.

After the learning procedure, weights data are fixed and saved to a file. Then transmit the weights to the Xilinx.

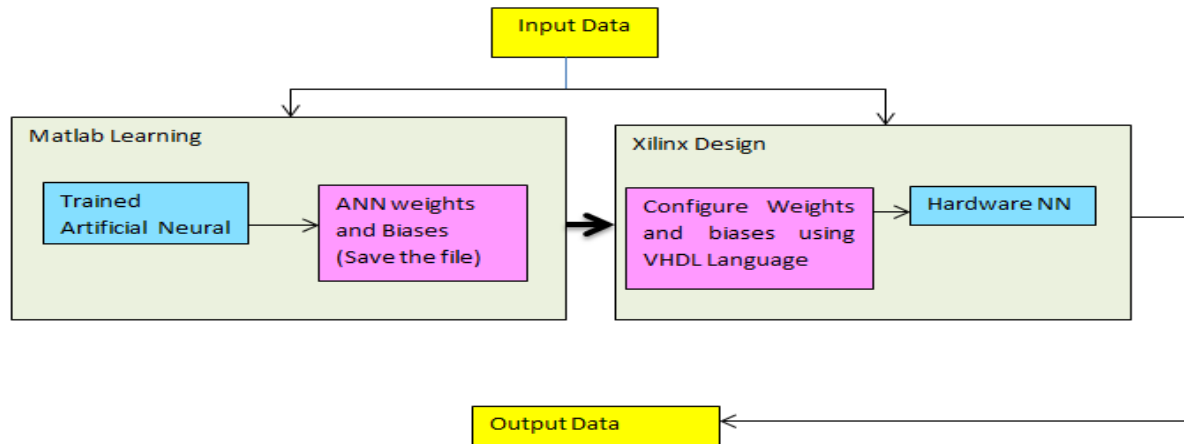


Figure 8.2: Design implementation of ANN on FPGA

This file, along with other VHDL coding is compiled, synthesized and implemented with Xilinx ISE software tools. Simulation results are visualized using ISIM and ModelSim. Finally the design is realized on a Xilinx Spartan 6 XC6SLX45 [9].

8.3 Design Implementation and Simulation Results

Design Algorithm of ANN implementation on FPGA Spartan 6 is described as follow:

Design Algorithm:

1. Decide the input parameters to the ANN.
2. Decide the output parameter from the ANN.
3. Calculate the input and output training pairs of ANN.
4. Train the ANN using the training pairs. After training, obtain the trained ANN.
5. Obtain the weights and biases from the trained ANN and input it to the VHDL code of ANN.
6. Execute the VHDL code for the decided input, weights and biases for ANN.
7. Observe the results using ModelSim OR ISIM.
8. Repeat the steps 3 to 7 for different inputs and verify the outputs.
9. Compare the results with MATLAB based ANN.
10. If the results are satisfactory then load the VHDL code in FPGA Chip.

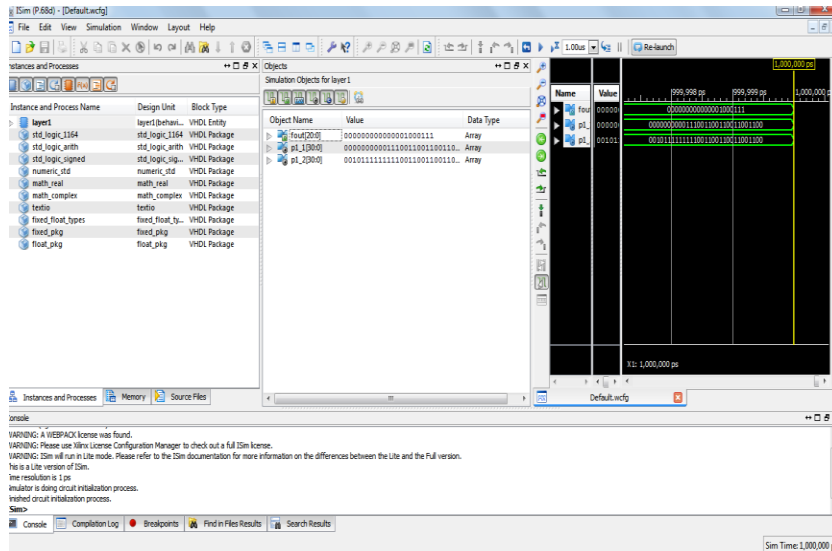


Figure 8.3: Output of ANN from ISIM Using VHDL code

Figure 8.3 shows the output of ANN model observed from the ISIM tool after simulating the FPGA program for Spartan 6 XC6SLX45 demo kit. Output shows waveforms of inputs p_1 and p_2 where two inputs are data rate and interarrival time deciding packet size by ANN model (discussed in chapter 7) and final output configure on FPGA.

The results of ANN using MATLAB and implemented in VHDL code are verified by calculating relative error. The following Table 8.1 compares the output of the complete neural network calculated using MATLAB program with ANN implemented using FPGA technique. The relative difference between Hardware and Software Implementation of ANN was compared in the terms of relative difference is shown in Figure 8.4.

Input Parameters		Packet Size Using Calculation	ANN based Packet size		Relative Difference (using MATLAB and using FPGA)
Interarrival Time (ms)	Data Rate (bits/sec)		Calculated using MATLAB	Designed using FPGA	
1.8	82	150	181	175.78	5.22
	164.473	300	322	312.5	9.5
	274.122	500	509	507.81	1.19
	383.771	700	695	693.35	1.6499
	493.421	900	880	878.91	1.09

Table 8.1 Comparison of ANN results of MATLAB and FPGA

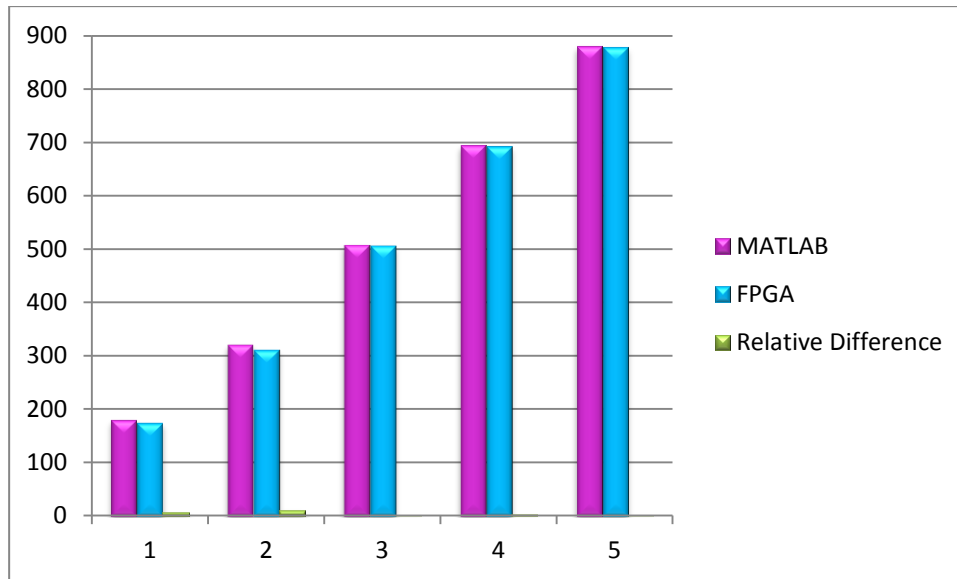


Figure 8.4: Relative Difference

8.4 Test-bed Hardware Implementation

The real time hardware implementation of ANN configuration on Spartan 6 kit setup is shown in below Figure 8.5 and 8.6.

The plus point of this Digilent Atlys Spartan 6 kit is easy way of programming the chip. The kit has Adapt system providing simplified programming interface and many additional features as described in the appendix. The Adept port is compatible with Xilinx's iMPACT programming software if the Digilent Plug-In for Xilinx Tools is installed on the host PC (download from the Digilent website's software section). Following Figure 8.7 shows the snap shot of programming a bit file on the Spartan 6 kit with Digilent Adept Software. Figure 8.8 shows the output on LED for given combination of the inputs Data Rate and Inter Arrival Time (p_1 and p_2).

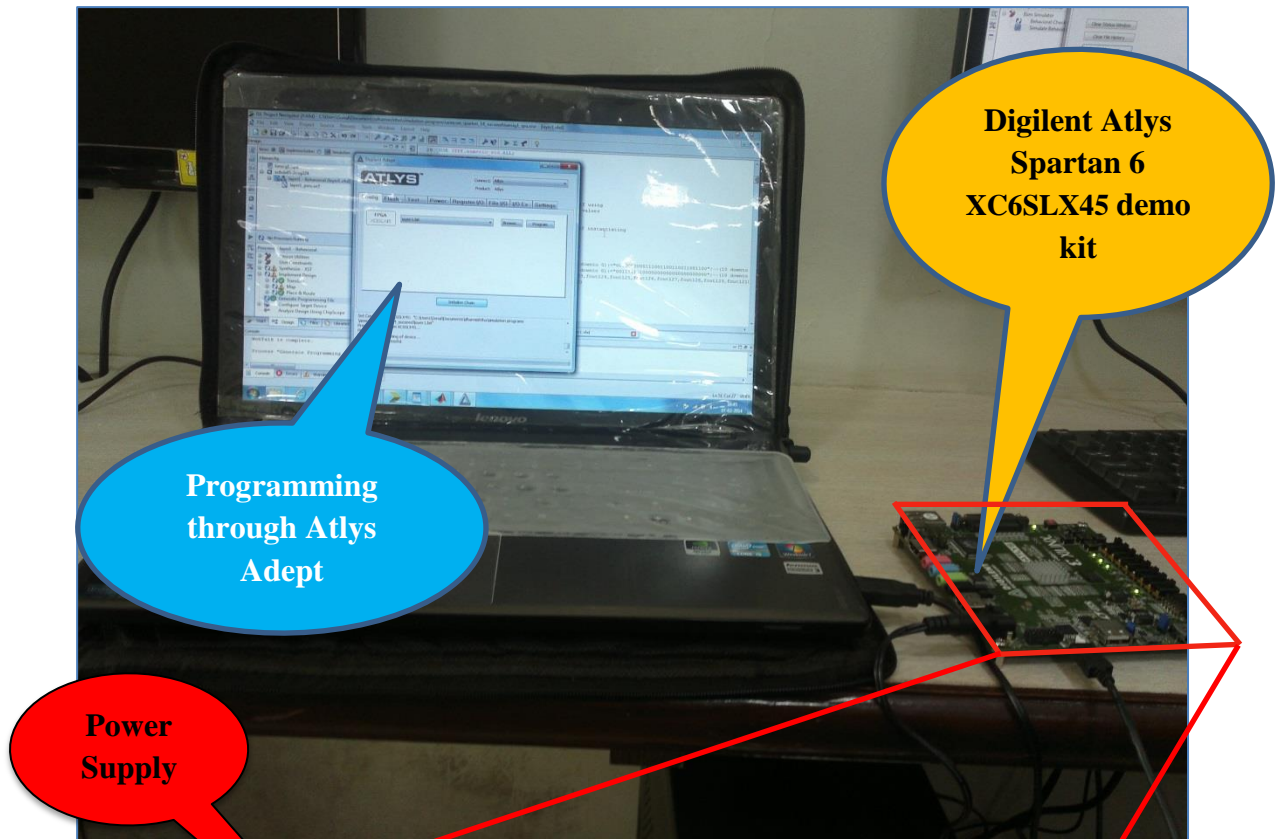


Figure 8.5 Hardware setup of ANN configuration on FPGA kit

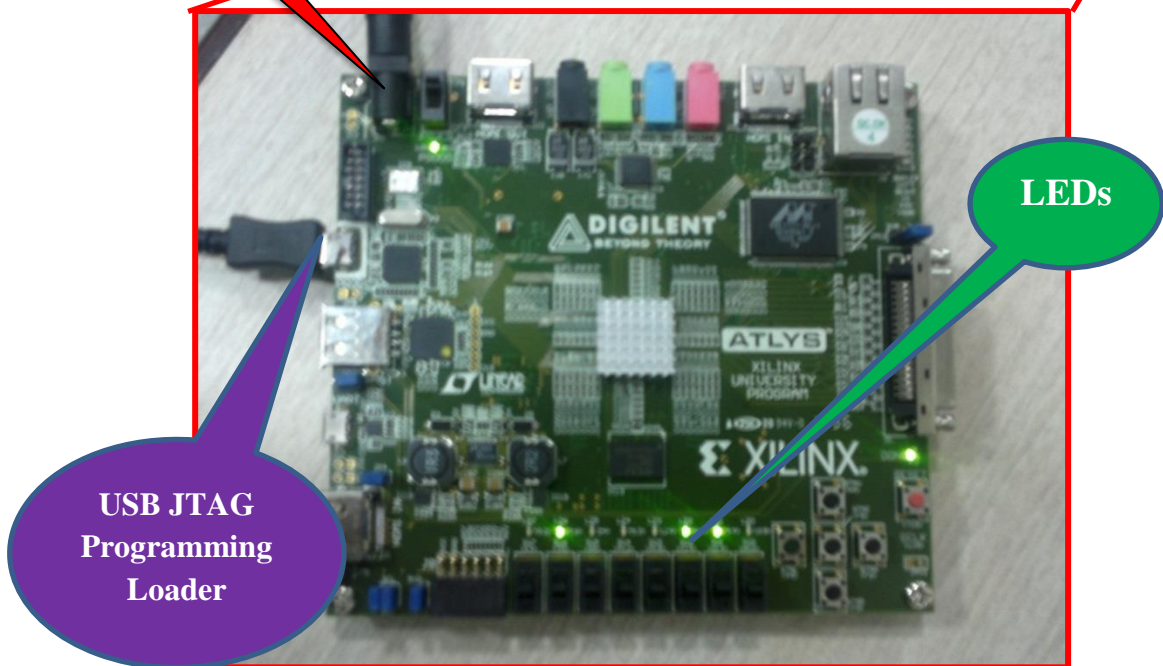


Figure 8.6: Enlarge view of Spartan 6 demo Kit

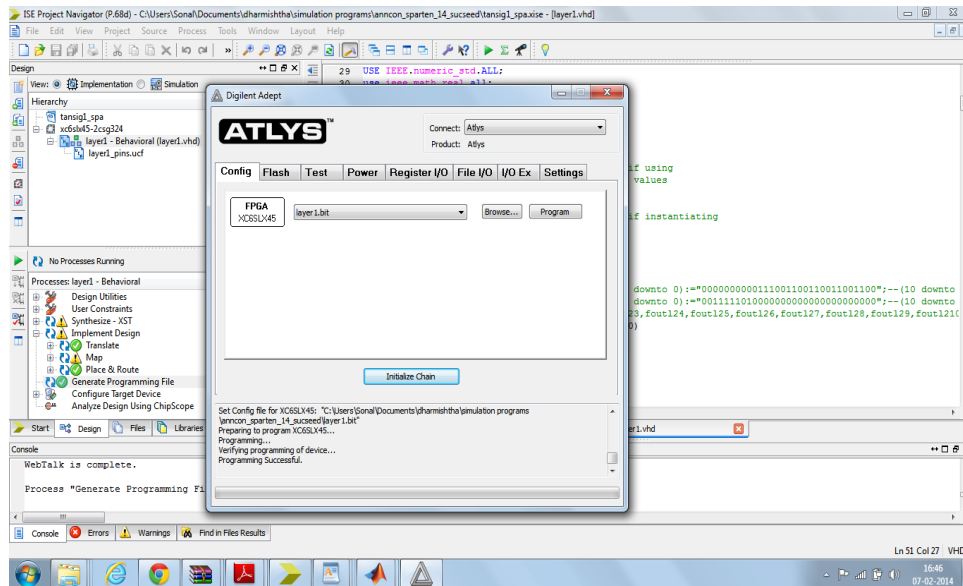


Figure 8.7: Snapshot of programming FPGA kit

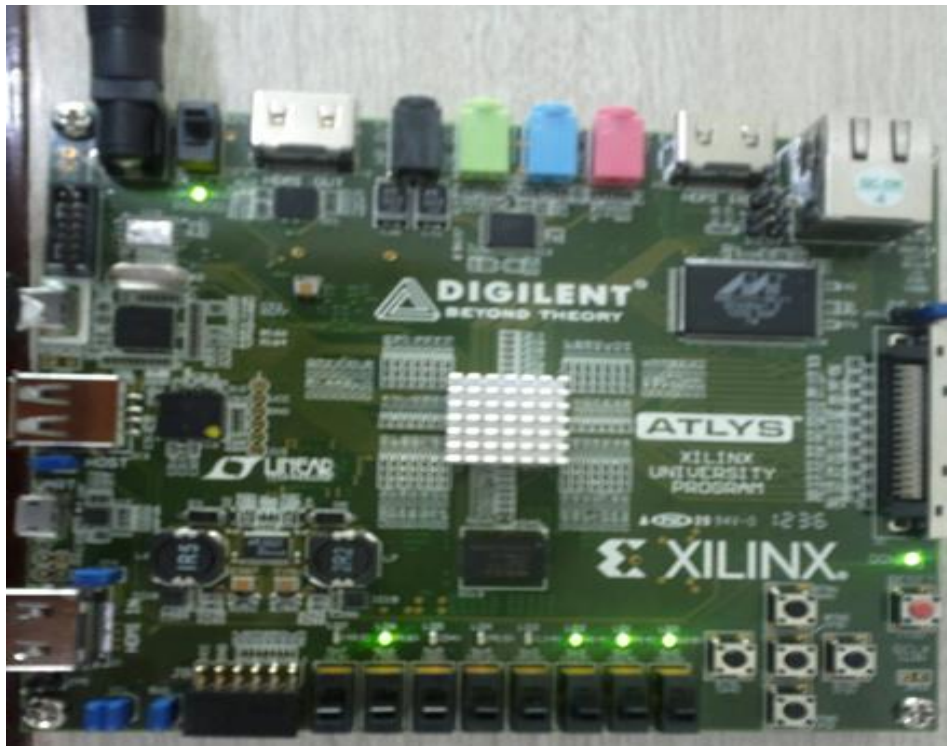


Figure 8.8: Output on LED for given combination of the inputs

SECTION B: WSN Hardware implementation

Current popular low-end wireless sensor network hardware is small sized, uses low cost Reduced Instruction Set Computer (RISC) microcontrollers and provides a small amount of program and data memory (about 100 kB). Mainly for status indication most boards integrate up to three LEDs. Many Companies such as Intel, Crossbow, Dust

Networks, Millennial Net, Arched Rock, Ember, and others manufacture Sensor network devices (motes).

The size of a wireless sensor nodes are usually varied from a shoe box size to the size of a gold coin [10]. The following Figure 8.9 shows what a typical wireless sensor node looks like. The future trend of WSN devices are going to become cheaper, smaller and longer energy lasting [11].



Figure 8.9: Typical wireless sensor nodes size [10]

The traditional wireless sensor (see Figure 8.10) consists of a communication device (e.g. radio transceiver/transmitter) for wireless communication, a microprocessor for processing data, sensing device (sensor board) for sensing of a physical or environment conditions, and power device (e.g. battery or solar panel) to provide the sensor nodes with the power needed [10,12, 13].

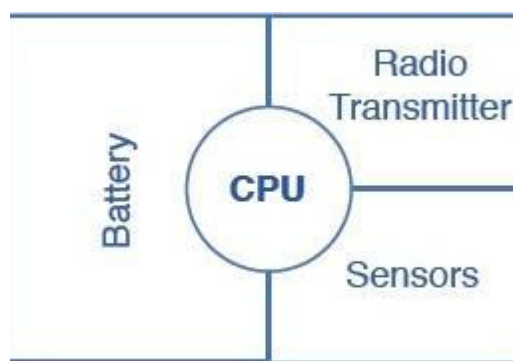


Figure 8.10: Wireless sensor node components [11]

According to the different applications of the sensor node, the function and quantity of the sensing device of the nodes are also different. It can detect temperature, humidity, acceleration, noise, light intensity, pressure, the size of moving objects, speed,

and many other physical phenomena in which the observer may be interested [10, 12, 14, 15].

The hardware features of the Mote Processor Radio (MPR) platforms and Mote Interface Boards (MIB) for network base stations and programming interfaces. It is intended for understanding and leveraging Crossbow's Smart Dust hardware design in real-world sensor network, smart RFID, and ubiquitous computing applications.

8.5 CROSSBOW MICAZ (MPR2400) MOTE Processor

MICAz [16] is the latest contribution to the Mica family evolution. Mica, released in 2001, was carefully designed to serve as a general platform for wireless sensor network research. Mica2, the successor to the Mica platform, was released one year later and corrected several of Mica's shortcomings. In 2004 MICAz was released and replaced the Chipcon CC1000 radio with the CC2420, an IEEE 802.15.4 compatible radio.

MICAz uses the Chipcon CC2420 radio in the 2.4 GHz band, a wideband radio with O-QPSK modulation with DSSS at 250kbs. The radio's higher data rates allows for shorter active periods and thereby reducing energy consumption. The CC2420 provides a number of hardware accelerators to achieve better performance. These include encryption and authentication, packet handling support, auto acknowledgments, and address decoding. The MICAz is capable of establishing and maintaining a multi-hop mesh network. The MICAz is used in this thesis for sensor-to-gateway communication.

Figure 8.11 below shows the general layout of the MICAz. The MicaZ specifications are described in [18].

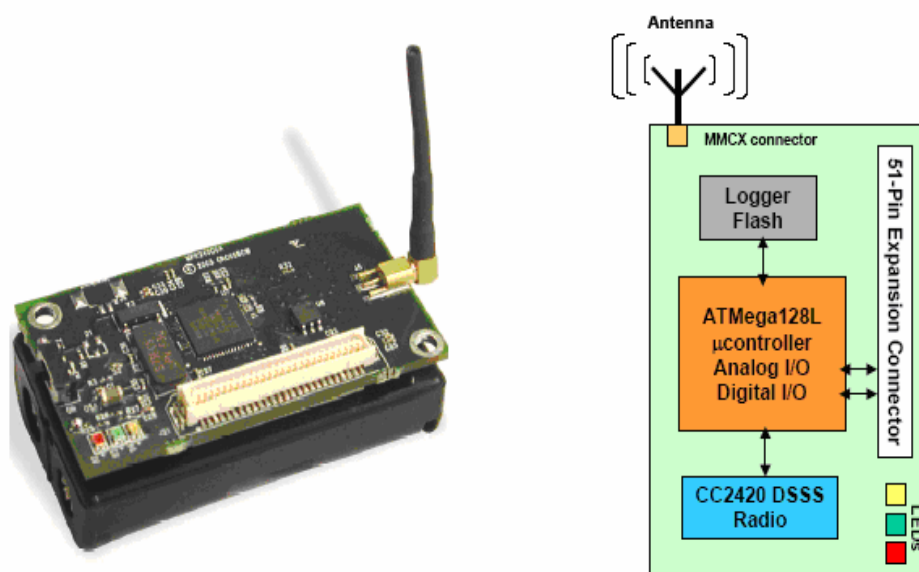


Figure 8.11: MICAz mote [courtesy Crossbow]

8.5.1 CC2420 radio transceiver

The CC2420 RF transceiver is mounted on the MPR2400 board for the purpose of wireless communication. It is a single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver designed for low power and low voltage wireless applications [17]. CC2420 includes a digital direct sequence spread spectrum (DSSS) baseband modem providing a spreading gain of 9 dB and an effective data rate of 250 kbps. The MicaZ's CC2420 radio can be tuned from 2.048 GHz to 3.072 GHz which includes the global Industrial, Scientific and Medical (ISM) band at 2.4 GHz. IEEE 802.15.4 channels are numbered from 11 (2.405 GHz) to 26 (2.480 GHz) each separated by 5 MHz.

The CC2420 provides one very important piece of metadata about received packets. This is received signal strength indicator (RSSI), which is a measurement of the power in dBm present in a received radio signal. It is calculated over the first eight symbols after the start of a packet frame. RSSI can also be sampled at other times, to detect the ambient RF energy. RF transmission power is programmable from 0 dBm to -25 dBm. Typically, the CC2420 consumes the current of 18.8 mA in the transmit mode and that of 17.4 mA in the receive mode and have a typical sensitivity of -95 dBm.

8.5.2 MIB520 USB interface board

The MIB520, shown in Figure 8.12, provides USB connectivity to the MICA family of Motes for communication and in-system programming. It supplies power to the devices through USB bus.

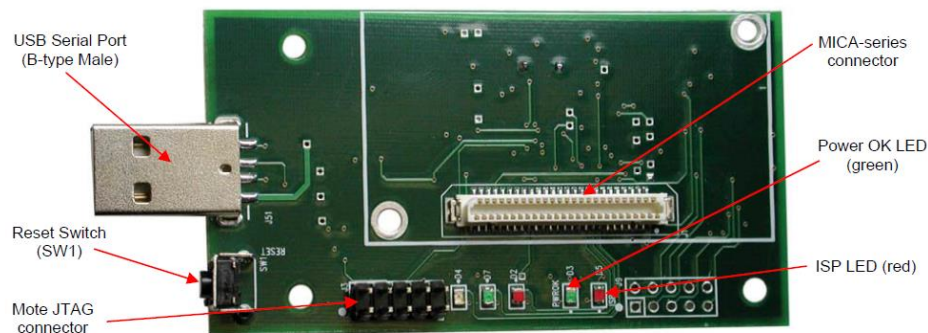


Figure 8.12: Photo of top view of an MIB520CB

The MIB 520 has an on-board in-system processor (ISP) – an ATmega16L to program the motes. Code is downloaded from a PC to the ISP through the USB port. Next the ISP programs the code into the mote. The mote which is attached to the MICA-series connector of the MIB520 is defined as the base station. It allows the aggregation of sensor network data onto a PC. Any MicaZ mote can function as a base station when it is

connected to the MIB520. Therefore, the MIB520 provides a fundamental serial/USB interface for both programming and data communications for any WSN.

8.5.3 MDA100CA/MDA100CB

MD100CA and MDA100CB (shown in Figure 8.13) have the same content except for some minor changes. The MDA100 series sensor boards have a precision thermistor, a light sensor/photocell, and general prototyping area. The prototyping area supports connection to all eight channels of the Mote's analog to digital converter (ADC0–7), both USART serial ports and the I²C digital communications bus. The prototyping area also has 45 unconnected holes that are used for breadboard of circuitry.



Figure 8.13: MDA 100CB

8.5.4 TinyOS

TinyOS [18] is an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture, which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools—all of which can be used as-is or be further refined for a custom application. TinyOS's event-driven execution model enables fine-grained power management yet allows the scheduling flexibility made necessary by the unpredictable nature of wireless communication and physical world interfaces.

TinyOS has a component-based programming model (codified by the nesC language). Like other operating systems, TinyOS organizes its software components into layers. The lower the layer the closer it is to the hardware; the higher the component, the closer it is to the application. A complete TinyOS application is a graph of components, each of which is an independent computational entity.

Components have three computational concepts: 1) commands, 2) events, and 3) tasks. Commands and events are mechanisms for inter-component communication, while tasks are used to express intra-component concurrency. A command is typically a request to a component to perform a service. A typical example is starting a sensor reading. By

comparison, an event would signal the completion of that service. Events may also be signalled asynchronously, for example, due to hardware interrupts or message arrival. From a traditional OS perspective, commands are analogous to downcalls and events to call backs. Commands and events cannot block. However, a request for a service is split-phase in that the request for service (the command) and the completion signal (the corresponding event) are decoupled. The command returns immediately and the event signals completion at a later time.

Rather than performing a computation immediately, commands and event handlers may post a task, a function executed by the TinyOS scheduler at a later time. This allows commands and events to be responsive, returning immediately while deferring extensive computation to tasks. While tasks may perform significant computation, their basic execution model is run-to-completion, rather than to run indefinitely; this allows tasks to be much lighter-weight than threads. Tasks represent internal concurrency within a component and may only access state information within that component. The TinyOS scheduler uses a non-preemptive, first in, first out (“FIFO”) scheduling policy. For more details on TinyOS and nesC programming concepts, refer to the “TinyOS/nesC Reference Manual” by Phil Levis included on the *MoteWorks* CD.

8.6 Software Description and Discussion

This section describes the software provided by the manufacturer for programming the motes.

8.6.1 Software Development Tools

MoteWorks™ [18] is the end-to-end enabling platform for the creation of wireless sensor networks. The optimized processor/radio hardware, industry-leading mesh networking software, gateway server middleware and client monitoring and management tools support the creation of reliable, easy-to-use wireless OEM solutions. OEMs are freed from the detailed complexities of designing wireless hardware and software enabling them to focus on adding unique differentiation to their applications while bringing innovative solutions to market quickly.

MoteWorks is provided with a set of software development tools for custom Mote applications, including custom sensor board drivers, sensor signal conditioning and processing and message handlers. MoteWorks includes an optimized cross-compiler for the target mote platform and an advanced editor for TinyOS application development. MoteWorks automatically installs and configures these development tools for quick set-up and rapid start of development.

Within the MoteWorks framework a minimum of five files will be placed in any application's directory:

1. Makefile
2. Makefile.component
3. Application's configuration written in nesC
4. Application's module written in nesC
5. README (optional)

Figure 8.14 (a) and (b) shows the method of executing .nc file and programming motes by writing the following command on Tools>shell.

Make micaz install,0 mib520,com10

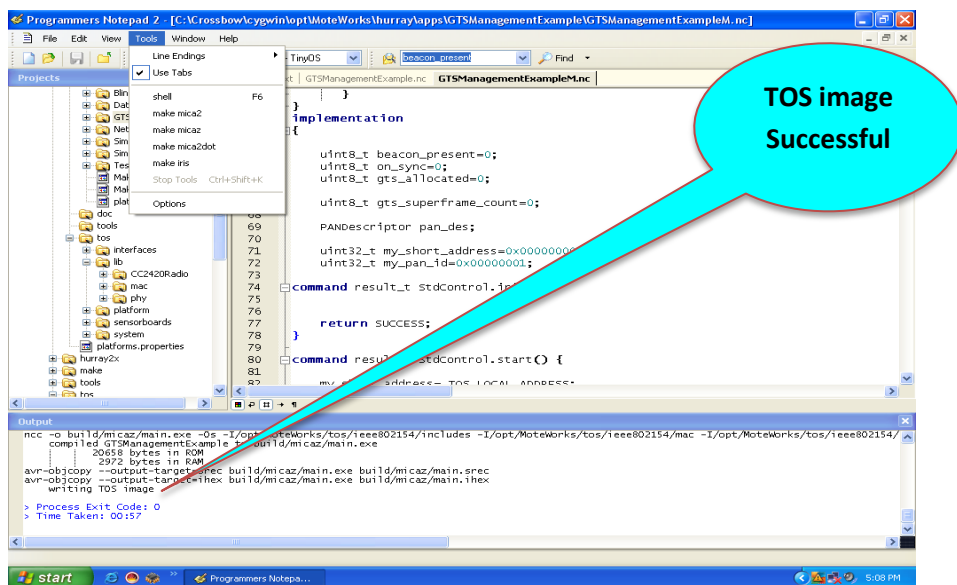


Figure 8.14 (a): Programming Environment of Motes

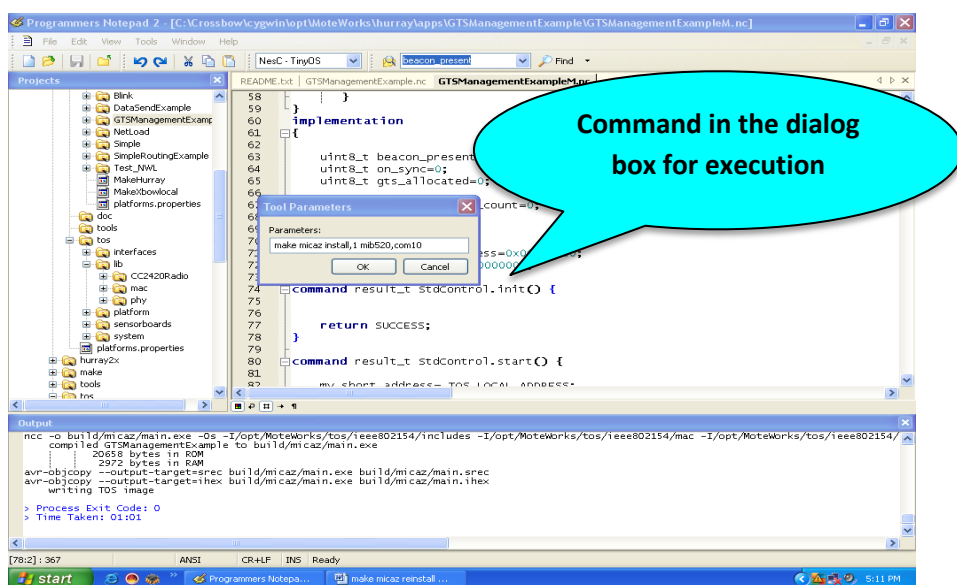


Figure 8.14 (b): Programming Environment of Motes

After the compilation has completed one should see “writing TOS image” as the last line in the Output window shown in Figure 8.14(a), otherwise it shows error in one of the files. After the successfully loading the programme in mote one can see the message “Uploading: flash” shown in Figure 8.15.

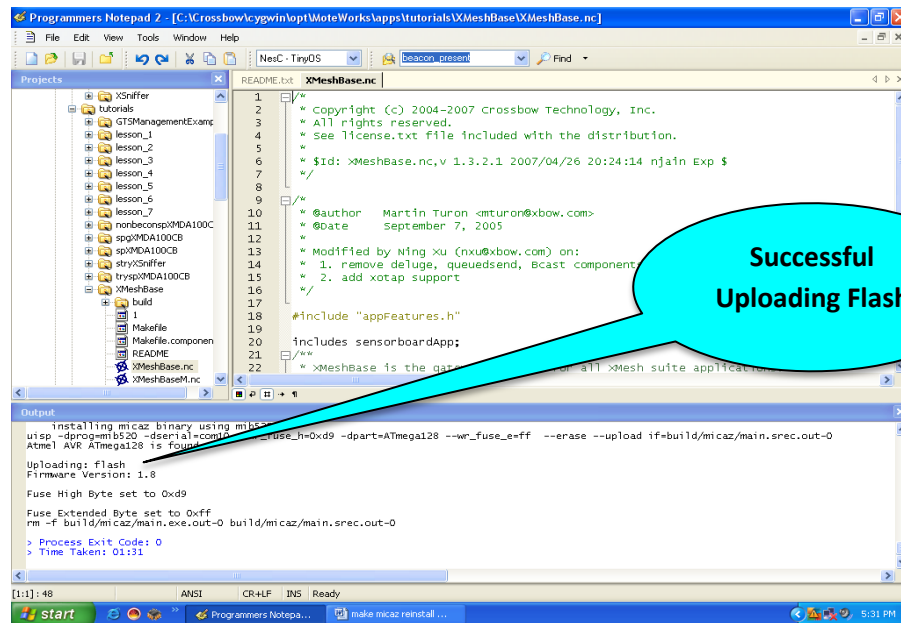


Figure 8.15: Snapshot of successful programming done in motes

8.6.2 MOTE-VIEW Functionalities

MoteView [19] is designed to be an interface between a user and a deployed network of wireless sensors. MoteView provides the tools to simplify deployment and monitoring. It also makes it easy to connect to a database, to analyze, and to graph sensor readings. The key function of the program is to monitor the communications between the gateway and the individual motes.

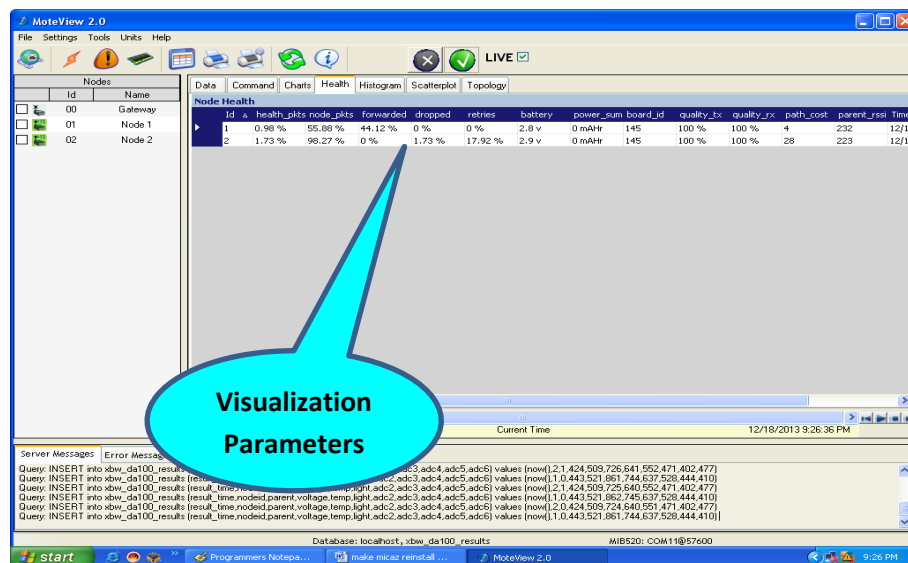


Figure 8.16: Screenshot of the database in Health view

The data can be displayed using the MoteView program. The color of the mote icons on the left hand side of the program's graphic user interface (GUI) indicates the overall health of the connection from the mote to the gateway. The green color indicates the signal is good and the latest signal received from the particular mote is current. Figure 8.16 shows the visualization of parameters. This screen can be accessed by selecting the Tools icon on the menu bar and selecting the Program Mote option. [19]

8.7 Test-bed Hardware Setup and Implementation

In this section, the entire network nodes are built on MICAz platform– MICAz XMDA100 WSN starter kit from CROSSBOW which includes three sensor nodes and one base station.

8.7.1 Program Sensor Nodes

This subsection explains test-bed on IEEE 802.15.4 WSN Star network operating in beacon enabled mode, with one PAN Coordinator and one End Device. Both real test-bed and the simulation will be set with the same initial parameters.

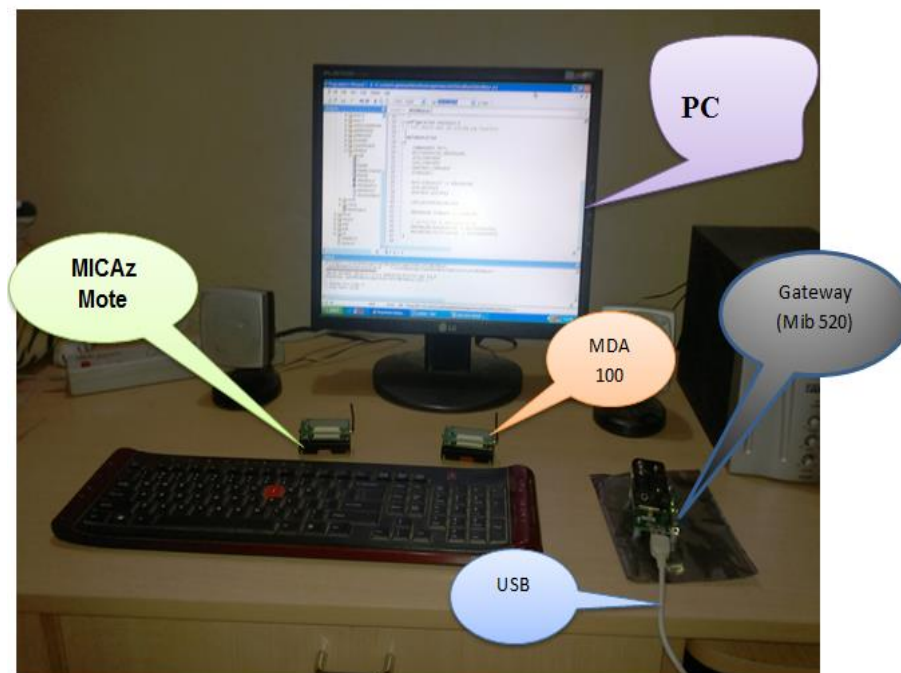


Figure 8.17: Experimental Test bed using MICAz Motes

Figure 8.17 depicts the experimental test-bed using MICAz motes. When the Coordinator node is turned on, the end node synchronizes with its beacon and starts transmitting data frames with the respective configurations.

Performance metrics were defined in order to evaluate the performance of the beacon enabled mode. These metrics are means of comparison between experimental and simulation results. The simulation and the experimental scenarios are depicted in Figure 8.18(a) and (b), respectively.

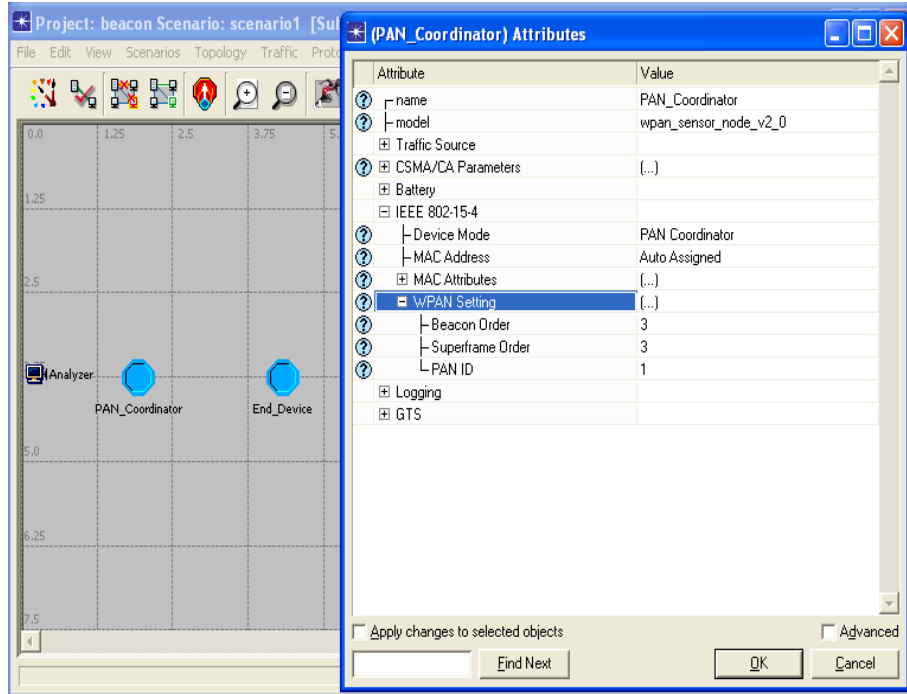


Figure 8.18(a): Simulation Test bed



Figure 8.18(b): MICAz Nodes & Gateway Setup

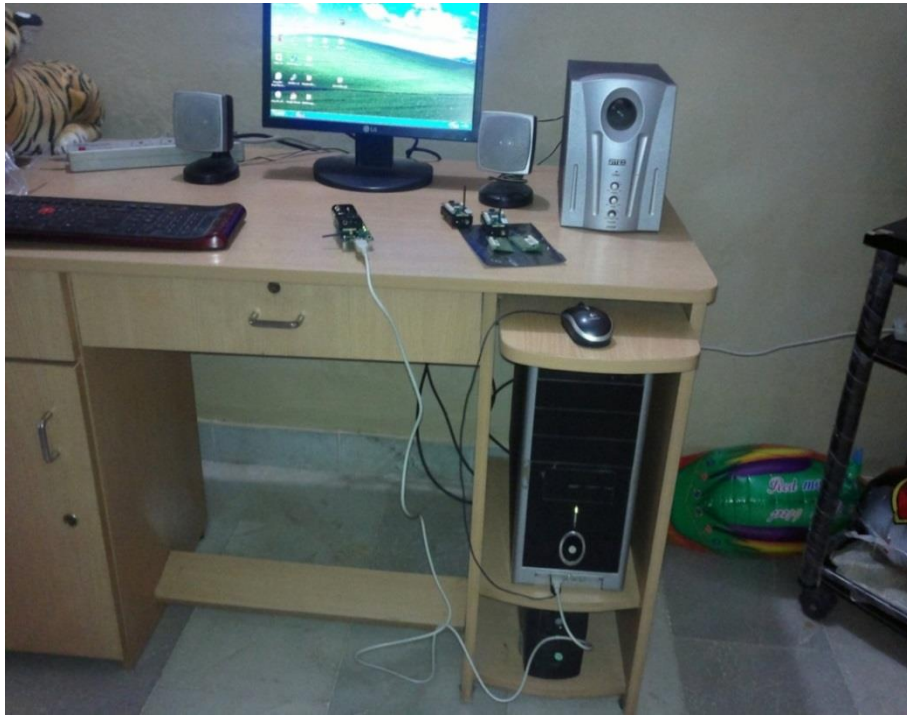


Figure 8.19: Snapshot of connecting mote on gateway through USB port

Figure 8.19 shows the snapshot of connecting mote on gateway and CPU USB port to programming the motes and Figure 8.20 shows the red LED on during successful uploading programme.



Figure 8.20: Snapshot of uploading programme

8.7.2 Experiment Results

Figure 8.21 shows the results of the node throughput of test-bed and simulation with same scenario. The nature of the graphs are similar for test-bed experiment and simulation i.e., for the traffic load, the node throughput decreases as the data rate increased. The results shown in test-bed experiments are considered in the four rounds experiments.

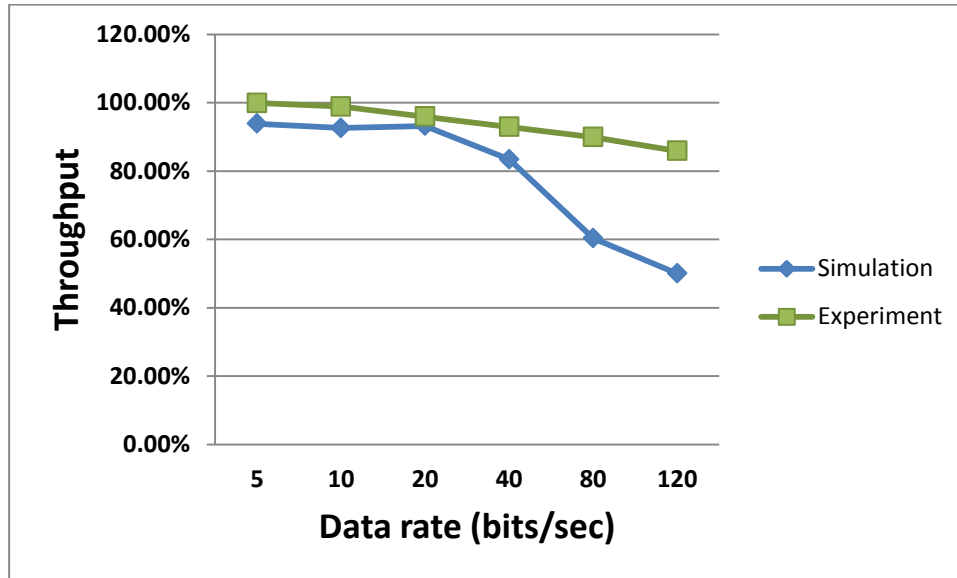


Figure 8.21: Test bed versus Simulation results

As it can be seen from the graph, nature of data rate versus throughput is same for both test-bed experiment and simulation; there is a small difference in both the results. The reasons cause the results distance between simulation and test-bed is that behaviour can be the increased number of failures due to higher medium congestion and the simulation default setting uses low RSSI and considers GTS (Guaranteed Time Slot) in the scenario.

Summary

This chapter describes FPGA hardware implementation of ANN configuration. Feed forward type Multilayer Perceptron (MLP) neural network with Tansig as an activation function is used to decide Packet size for given input parameters data rate and inter arrival time. Result comparison of the FPGA implementation of ANN with the Matlab implementation is done by calculating relative error. In the second section of this chapter implements real time WSN using MICAz motes with same scenario in simulation and comparison is done to get conclusion that simulation and experiment gives same behavior for same environment.



Chapter 9



Conclusion & Future Scope



This chapter describes the conclusions about the results and summarisation of research work carried out. Also some future scope suggested here to extend the project for further real time implementations.

Simulation is an important methodology to develop, investigate and evaluate the network systems. The simulation study was carried out using software simulation tools such as: MATLAB/SIMULINK, True Time, and OPNET. The parametric optimization is considered to evaluate the performance of WSN based on performance metrics.

OPNET simulation tool was used to evaluate the performance of the Wireless and Wired Network in terms of different number of users, traffics. For high traffic and users wired network outperforms than Wireless network due to the transmission limit, SNR (signal to noise) and bandwidth of the received signal. So to improve the overall performance of the system it is better to use hybrid network which is the combination of both wired and wireless network.

To communicate nodes with each other in ad-hoc manner there are various constraints affects to the performance of the WSN in order to extend the communication range and maintain network scalability. The main WSN limitations are battery capacity (power consumption), bandwidth underutilization and computing power. Hence, packet routing techniques must be applied to provide long-range and large-scale communication in WSNs.

As signal transmission power utilized by nodes of WSN increases, the performance of WSN improves in terms of the communication range. So the nodes can communicate with less propagation delay and can recover the information from all the visiting points. All the nodes of WSN are battery operated nodes and it is not suitable to increase the power to improve the performance of network. The suitable algorithm can be used to determine the optimized value of the transmission power for each node so that less power can be utilized by nodes to exchange the information. Also soft computing techniques can be applied to determine the optimized value of the transmission power.

Sustaining a route from a source to a destination may consume more bandwidth and power than is required to support the data traffic flow. In order to optimize the communication, it is important to know the characteristics of the WSN topological structure. Parametric evaluation for Topological structure like Tree, Mesh and Star of WSN has been carried out and its performance was analyzed.

IEEE 802.15.4 Standard for WSN is used in beacon enabled mode to transfer the data under Superframe structure to improve the performance of it. For the Contention Free Period GTS Mechanism is analyzed for WSN. GTS (local) throughput and Global throughput for the WSN beacon enabled mode are evaluated for GTS Mechanism using IEEE 802.15.4 OPNET simulation model in terms of Superframe Order (SO) and Beacon Order (BO). Selection of Superframe Order must be done carefully to ensure that the GTS in a superframe can accommodate at least one packet size and higher SO values i.e., greater than 7 are not supported by WSN application. The maximum utilization of the allocated GTS is achieved for superframe orders $SO = [2, 3, 4, \text{ and } 5]$. If 100% duty cycle is not considered ($SO < BO$) then inactive period increases and throughput decreases which is influenced by processing, transmitting, propagation, and queuing delays. The throughput can be increased by decreasing wasted bandwidth which occurs because of a fraction of the slot can be used during transmission of data and remaining bandwidth is wasted in every GTS slot in every superframe. This can be resolved by splitting the slots which make superframe structure with 32 slots. This method can be applicable only for small data transmission which can accommodate with this new slot duration. If data is large which cannot accommodate with this modified slot duration then modified method may not give optimum solution.

To reduce the wasted Bandwidth and to increase the GTS throughput, packet size has been decided by Soft Computing Techniques ANN and ANFIS. Both are trained to determine the packet size adaptively depending upon the input parameters Data Rate and Inter Arrival Time and give the optimum Packet size to get optimum GTS Throughput based on network situation and the parameters.

Hardware implementation of Soft Computing Technique ANN is implemented on FPGA using Spartan 6 XC6SLX45 in VHDL code. The relative difference of Hardware (FPGA) and Software (MATLAB) implementation is negligible. The error can be further minimized by considering the more number of bits to represent weights and biases of ANN and accuracy also can be increased with the cost of memory.

The performance of WSN is implemented on Hardware using MICAz motes. Throughput of WSN of Hardware implementation and Simulation Implementation is evaluated gives almost similar behavior.

Future work will explore some of the research directions pointed out in the thesis so far

- ✂ The optimization of routing communication and signal reach parameters can be done using soft computing technique.
- ✂ Here we have considered star topology; other topology can be optimized using different algorithms. The future work involves testing our work under different simulation models using a wider range of protocol parameters and more complex network topologies.
- ✂ There is also a room for improvements using IEEE 802.15.4 slotted CSMA/CA for time-critical events, priority-based delay mitigation and special GTS allocation mechanisms for time sensitive networks.
- ✂ Genetic Algorithm can be used to tune the weights and biases of ANN and ANFIS.



Chapter 10



Bibliography



Chapter 1

- [1] IEEE Standard 802.15.4-2006, September, “Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)”, IEEE, 2006. [Online].Available: <http://www.ieee802.org/15>.
- [2] Zigbee-Alliance, “ZigBee Specification,” <http://www.zigbee.org/>
- [3] V. Thirunavukkarasu, M. Kannan, “Throughput, End-to-End Delay and Utilization Analysis of Beacon Enabled and Non-Beacon Enabled WSN”, European Journal of Science Research, ISSN 1450-216X Vol.81 No.2 (2012), pp.196-209.
- [4] J. Mišić, and V. B. Mišić, “Duty Cycle Management in Sensor Networks Based on 802.15.4 Beacon Enabled MAC”, Ad Hoc and Sensor Wireless Networks Journal, Old City Publishing, 1(3):207-233, 2005.
- [5] J. Mišić and V. B. Mišić, “Access Delay and Throughput for Uplink Transmissions in IEEE 802.15.4 PAN”, Elsevier Computer Communications Journal, 28(10):1152-1166, June 2005.
- [6] J. Mišić, S. Shafi, and V. B. Mišić, “The Impact of MAC Parameters on the Performance of 802.15.4 PAN”, Elsevier Ad hoc Networks Journal, 3(5):509–528, 2005.
- [7] J. Zheng, M. L. Lee, “A Comprehensive Performance Study of IEEE 802.15”, IEEE Press Book, 2004.
- [8] Mishra, Amitabh and Na, Chewoo and Rosenburgh, Dwayne, “On Scheduling Guaranteed Time Slots for Time Sensitive Transactions in IEEE 802.15.4 Networks”, In Military Communications Conference, MILCOM 2007. IEEE, pp. 1-7, 2007.
- [9] A. Koubaa, M. Alves, and E. Tovar, “i-GAME: An Implicit GTS Allocation Mechanism in IEEE 802.15.4,” Euromicro Conference on Real-Time Systems (ECRTS’06), Jul. 2006.
- [10] Koubaa, A. and Alves, M. and Tovar, “GTS allocation analysis in IEEE 802.15.4 for real-time wireless sensor networks” In Parallel and Distributed Processing Symposium, IPDPS, 20th International, 8 pp., 2006.
- [11] Feng Chen and Talanis, T. and German, R. and Dressler, “Real-time enabled IEEE 802.15.4 sensor networks in industrial automation” In Industrial Embedded Systems, SIES '09, IEEE International Symposium on, pp. 136-139, 2009.

- [12] Pangun Park and Fischione, C. and Johansson, “Performance Analysis of GTS Allocation in Beacon Enabled IEEE 802.15.4”. In Sensor, Mesh and Ad Hoc Communications and Networks, SECON '09. 6th Annual IEEE Communications Society Conference on, pp. 1-9, 2009.
- [13] Hassan, M.N. and Stewart, R., “Analysis of buffer size dimensioning in GTS enabled IEEE 802.15.4 WSN for real-time applications”, Int. J. Mobile Network Design and Innovation, Vol. 3, No. 4, pp.231–238, 2011.
- [14] Haykin S., “Neural Networks: A Comprehensive Foundation”, Macmillian College Publishing: New York, NY, USA, 1994.
- [15] J. S. R. Jang, “ANFIS: Adaptive-Network-based Fuzzy Inference System”, IEEE Transactions on Systems, Man and Cybernetics, Vol. 23, pp. 665-685, 1993.
- [16] Y. J. Mon, “Airbag Controller Designed by Adaptive-Network-based Fuzzy Inference System (ANFIS), Fuzzy Sets and Systems”, Vol. 158, pp. 2706-2714, 2007.
- [17] S. Kurnaz and O. Çetin “Autonomous Navigation and Landing Tasks for Fixed Wing Small Unmanned Aerial Vehicles”, ActaPolytechnicaHungari-ca, Vol. 7, No. 1, pp. 87-102, 2010.
- [18] User Guide: Neural Network Tool Box (use with MATLAB) - The Mathworks Inc.
- [19] Petr Jurčík, Anis Koubâa, “The IEEE 802.15.4 OPNET Simulation Model: Reference Guide v2.0”, www.open-zb.net, IPP-HURRAY Technical Report, (TR-070509), May 2007.
- [20] OPNET Technologies, Inc., Opnet Modeler Wireless Suite –ver. 11.5A, <http://www.opnet.com>.
- [21] André Ribeiro e Cunha, “On the use of IEEE 802.15.4/ZigBee as federating communication protocols for Wireless Sensor Networks”, Ph.D. dissertation, Polytechnic Institute of Porto (ISEP-IPP), Portugal, July 2007.
- [22] Ember, www.ember.com, 2007.
- [23] Ember, “EM250 Single-Chip ZigBee/802.15.4 Solution”, Datasheet http://www.ember.com/products_zigbee_chips_e250.html, 2006.
- [24] Ember, “EM260 ZigBee/802.15.4 Network Processor”, Datasheet http://www.ember.com/products_zigbee_chips_e260.html, 2006.
- [25] Freescale semiconductor, www.freescale.com, 2007.
- [26] Freescale, “MC13192 2.4 GHz Low Power Transceiver for the IEEE® 802.15.4 Standard”, Technical Datasheet, www.freescale.com, 2007.

- [27] Freescale, “MC13201 2.4 GHz Low Power Transceiver for the IEEE® 802.15.4 Standard”, Technical Datasheet, www.freescale.com, 2007.
- [28] Integration, “IA OEM-DAUB1 2400 - IEEE 802.15.4/ZigBee USB Dongle”, www.integration.com, 2006.
- [29] Integration Associates, www.integration.com, 2007.
- [30] Texas Instruments, “Z-Stack”, <http://focus.ti.com/docs/toolsw/folders/print/z-stack.html>, 2007.
- [31] Texas Instruments, “CC2431 System-on-Chip for 2.4 GHz ZigBee/ IEEE 802.15.4 with Location Engine”, Datasheet, <http://focus.ti.com/docs/prod/folders/print/cc2431.html>, 2007.
- [32] Prof. Satish K. Shah, Ms. Sonal J. Rane, Ms. Dharmistha D Vishwakarma, “Performance Evaluation of Wired and Wireless Local Area Networks“ in the International Journal of Engineering Research and Development ISSN: 2278-067X, Volume 1, Issue 11, PP.43-48, July 2012.
- [33] Website: WWW.MATHWORKS.COM, The Mathworks Inc.
- [34] Alberto Cardoso, Sergio Santos, Amancio Santos and Paulo Gil “Simulation Platform for Wireless Sensor Networks based on the TrueTime Toolbox”.
- [35] Prof. Satish K. Shah, Ms. Sonal J. Rane, Ms. Dharmistha D Vishwakarma, “Simulation Study of Behaviour of Wireless Motes With Reference To Parametric Variation “ in the International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 1, Issue 2, pp:91-95 ISSN 2278 – 8875, August 2012.
- [36] Francesca Cuomo; Emanuele Cipollone; Anna Abbagnale; “Performance analysis of IEEE 802.15.4 wireless sensor networks: An insight into the topology formation process”, Comput. Netw. (2009). Doi: 10.1016/j.comnet.2009.07.016.
- [37] Prof. Satish K. Shah, Ms. Sonal J. Rane, Ms. Dharmistha D Vishwakarma, “Analytical Approach for Performance of Wireless Sensor Networks “in the International Journal of Electronics and Computer Science Engineering ,PP.1877-1884 , ISSN- 2277-1956.
- [38] Prof. Satish K. Shah, Ms. Sonal J. Rane, Ms. Dharmistha D Vishwakarma, “Throughput Optimization in IEEE 802.15.4 Using GTS Mechanism” International Journal of Latest Research in Science and Technology ISSN (Online):2278-5299 Volume 2, Issue 1: January-February 2013.

- [39] Prof. Satish K. Shah, Ms. Sonal J. Rane, Ms. Dharmistha D Vishwakarma, “Soft Computing Technique Based Throughput Optimization of GTS mechanism for IEEE 802.15.4 Standard” in the journal of Soft Computing and Software Engineering 2013.

Chapter 2

- [1] Simon Haykin, Communication Systems, 3rd ed. (Wiley).
- [2] Dr. R.K.Bansal,Vikas Gupta,Rahul Malhotra, “Performance analysis of wired and wireless LAN using soft computing techniques-A review”, Global Journal of Computer Science and Technology, Vol. 10 Issue 8 Ver.1.0 September 2010.
- [3] Jia Wang and Srinivasan Keshav, “Efficient and Accurate Ethernet Simulation”, Proc. Of the 24th Conference on Local Computer Networks (LCN'99), pp. 182-191, 1999.
- [4] Ikram Ud Din, Saeed Mahfooz and Muhammad Adnan, “Performance Evaluation of Different Ethernet LANs Connected by Switches and Hubs”, European Journal of Scientific Research, vol. 37 No. 3, pp. 461-470, 2009.
- [5] Regis J. Bates, “Wireless Networked Communications”, McGraw- Hill, 1994.
- [6] Mohammad Hussain Ali and Manal Kadhim Odah, “Simulation Study of 802.11b DCF using OPNET Simulator”, Eng. & Tech. Journal, vol.27,No6,2009,pp:1108-1117,2009.
- [7] M. Brownfield, K. Mehrjoo, A. Fayez, and N. Davis, “Wireless Sensor Network Energy-Adaptive MAC Protocol,” IEEE Consumer Communications and Networking Conference 2006 (CCNC 2006), Volume 2, pp. 778-782, January 2006.
- [8] Federal Communications Commission, “Operation within the bands 902-928 MHz, 2400-2483.5 MHz, and 5725-5850 MHz,” Title 47, Vol. 1, U.S. Government Printing Office via GPO Access, October 2001.
- [9] Eng. Tamer Mohamed Samir Khattab, “ Performance Analysis of Wireless Local Area Networks (WLANs) “ M.Sc. thesis, Faculty Of Engineering, Cairo University Giza, Egypt,2000.
- [10] IEEE 802.11 WG, Reference number ISO/IEC 8802- 11:1999(E) IEEE Std 802.11, 1999 edition, International Standard [for] Information Technology - Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, 1999.

- [11] IEEE 802.3-2002 IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications.
- [12] A. Athanasopoulos, S. Giannoulis, C. Antonopoulos, A. Prayati, E. Topalis, S. Koubias, "Performance evaluation of hybrid wired/wireless LANs for multimedia-like data traffic ", Applied Electronics Laboratory, Department of Electrical & Computer Engineering University of Patras, Rio Campus, Greece.
- [13] Hongwei Zhang, "Wireless Networking: An Introduction" available at <http://www.cs.wayne.edu/~hzhang>.
- [14] Peter T. Davis and Craig R. McGuffin, "Wireless Local Area Networks : Technology Issues and Strategies", McGraw-Hill, 1995
- [15] M. Brownfield, A. Fayez, and N. Davis, "Wireless Sensor Network Radio Power Management," In OPNETWORK 2005, August 2005.
- [16] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network",; in IEEE Personal Communications, Vol. 7, Issue 5, pp. 16-27, October 2000.
- [17] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring", ACM International Workshop on Wireless Sensor Networks and Applications, pp. 88-97, Sep. 2002,.
- [18] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler, "An analysis of a large-scale habitat monitoring application," In Proceedings of the Second ACM Conference on Embedded Networked Systems (SenSys), November 2004.
- [19] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," In Proceedings of the First European Workshop on Sensor Networks (EWSN), January 2004.
- [20] Achir M. and Ouvry L., "Power consumption prediction in wireless sensor networks", ITC Specialist Seminar on Performance Evaluation of Wireless and Mobile Systems, Antwerp, Belgium, 2004.
- [21] Jones C.E., Sivalingam K.M., Agrawal P. and Chen J.C., "A survey of energy efficient network protocols for wireless networks", Wireless Networks 7(4), 343–358, 2001.

- [22] Ye W., Heidemann J. and Estrin D., “Medium access control with coordinated adaptive sleeping for wireless sensor networks”, *ACM/IEEE Transactions on Networking* 12(3), 493–506, 2004.
- [23] Jung E.S. and Vaidya N.H., “A power control MAC protocol for ad hoc networks”, *Wireless Networks* 11(1–2), 55–66, 2005.
- [24] Van Dam T. and Langendoen K., “An adaptive energy-efficient MAC protocol for wireless sensor networks”, In *Proc. ACM SenSys03*, pp. 171–180, Los Angeles, CA, 2003.
- [25] Akan O.B. and Akyildiz I.F., “ESRT: event-to-sink reliable transport in wireless sensor networks”, *ACM/IEEE Transactions on Networking* 13(5), 1003–1016, 2005.
- [26] Akyildiz I.F., Su W, Sankarasubramaniam Y. and Cayirci E., “Wireless sensor networks: A survey”, *Computer Networks* 38, 393–422, 2002.
- [27] Intanagonwiwat C., Govindan R., Estrin D., Heidemann J. and Silva F., “Directed Diffusion for Wireless Sensor Networking”, *ACM/IEEE Transactions on Networking* 11(1), 2–16, 2003.
- [28] Callaway, Jr. E.H., “Wireless Sensor Networks, Architecture and Protocols”, Auerbach Publications, Boca Raton, FL, 2004.

Chapter 3

- [1] Harsh Sundani, Haoyue Li, Vijay K. Devabhaktuni, Mansoor Alam, & Prabir Bhattacharya,” Wireless Sensor Network Simulators A Survey and Comparisons “ , *International Journal Of Computer Networks (IJCN)*, Volume (2) : Issue (5) 249.
- [2] Fei Yu,” A Survey of Wireless Sensor Network Simulation Tools”, <http://www1.cse.wustl.edu/~jain/cse567-11/ftp/sensor/index.html>.
- [3] ZhengYi Guan (Alex), “A Reliability Evaluation of Wireless Sensor Network Simulator: Simulation vs. Test bed”, UNITEC New Zealand, 2011.
- [4] I.F. Akyildiz and W. Su and Y. Sankarasubramaniam and E. Cayirci, “A Survey on Sensor Networks,” *IEEE Communication Magazine*, vol. 40, no. 8, pp. 102-116, Aug. 2002.
- [5] Olivares, T., Tirado, P. J., & Orozco-Barbosa, L., “Simulation of power-aware wireless sensor network architectures”, Paper presented at the Proceedings of the ACM international workshop on Performance monitoring, measurement, and evaluation of heterogeneous wireless and wired networks, Terromolinos, Spain, 2006.

- [6] Kulakowski, P., "WIRELESS SENSOR NETWORKS: TECHNOLOGY, PROTOCOLS, AND APPLICATIONS". [Book Review]. IEEE Communications Magazine, 46(6), 42-44, 2008.
- [7] Park, S., Savvides, A., & Srivastava, M. B., "Simulating networks of wireless sensors", Paper presented at the Proceedings of the 33rd conference on winter simulation, Arlington, Virginia, 2001.
- [8] Glaser, J., Weber, D., Madani, S. A., & Mahlknecht, S., "Power Aware Simulation Framework for Wireless Sensor Networks and Nodes" [Article]. EURASIP Journal on Embedded Systems, 1-16. doi: 10.1155/2008/369178.
- [9] Haiming, C., Li, C. U. I., Changcheng, H., & He, Z. H. U. EasiSim, "A Scalable Simulator for Wireless Sensor Networks" [Article]. Wireless Sensor Network, 1(5), 467-474. doi: 10.4236/wsn.2009.15056.
- [10] NS-2. The Network Simulator - ns-2 Retrieved 15/05/2011, from <http://www.isi.edu/nsnam/ns/>
- [11] Stevens C., Lyons C., Hendrych R., Carbajo R. S., Huggard M., & Goldrick C. M, "Simulating Mobility in WSNs: Bridging the Gap between ns-2 and TOSSIM 2.x", Paper presented at the Proceedings of the 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications.
- [12] T. Issariyakul and E. Hossain, "Introduction to network simulator ns2", Springer, Nov. 2008.
- [13] Abdelrahman Abuarqoub, Fayez Al-Fayez, Tariq Alsboui, Mohammad Hammoudeh, Andrew Nisbet, "Simulation Issues in Wireless Sensor Networks: A Survey", SENSORCOMM 2012: The Sixth International Conference on Sensor Technologies and Applications.
- [14] Shu L., Wu C. Zhang Y., Chen J., Wang L., & Hauswirth M., "NetTopo: beyond simulator and visualizer for wireless sensor networks". SIGBED Rev., 5(3), 1-8. doi: 10.1145/1534490.1534492, 2008.
- [15] Nsnam, "NS-3," 2011; from <http://www.nsnam.org/>.
- [16] Parallel Computing Laboratory at UCLA. (2010, May) GloMoSim. [Online]. Available: <http://pcl.cs.ucla.edu/projects/glomosim/>
- [17] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla, "GloMoSim: A scalable network simulation environment," UCLA Computer Science Department Technical Report, vol. 990027, p. 213, 1999.

- [18] J. Nuevo, "A Comprehensible GloMoSim Tutorial. Quebec, Canada, March 2004," 2006.
- [19] Parallel Computing Laboratory at UCLA. (2010, May) Parsec. [Online]. Available: <http://pcl.cs.ucla.edu/projects/parsec/>
- [20] D. Curren, "A survey of simulation in sensor networks," 2005.
- [21] S.N. Technologies, "QualNet Simulator," from <http://www.scalable-networks.com/products/qualnet/>.
- [22] TinyOS, Available from: <http://www.tinyos.net>
- [23] Levis P., Lee N., Welsh M., & Culler D., "TOSSIM: accurate and scalable simulation of entire TinyOS applications", Paper presented at the Proceedings of the 1st international conference on Embedded networked sensor systems, Los Angeles, California, USA, 2003.
- [24] "Python", URL: <http://en.wikipedia.org/wiki/Python>, Description: an introduction of Python in wiki webpage.
- [25] B. Titzer, D. Lee, and J. Palsberg, "Avrora: Scalable Sensor Network Simulation with Precise Timing," in Proceedings of IPSN'05, Fourth International Conference on Information Processing in Sensor Networks, (Los Angeles, USA), 25–27 April 2005.
- [26] Bartosz Musznicki and Piotr Zwierzykowski, "Survey of Simulators for Wireless Sensor Networks", Vol. 5, No. 3, September, 2012.
- [27] Geoff V. Merrett, Neil M. White, Nick R. Harris and Bashir M. Al-Hashimi, "Energy-Aware Simulation for Wireless Sensor Networks", Electronic Systems and Devices Group, School of Electronics and Computer Science University of Southampton Southampton, UK.
- [28] "Avrora", URL: <http://compilers.cs.ucla.edu/avrora/>, Description: a webpage introduced Avrora.
- [29] Lei Shu, Chun Wu, Yan Zhang, Jiming Chen, Lei Wang, Manfred Hauswirth, "NetTopo: Beyond Simulator and Visualizer for Wireless Sensor Networks", Future Generation Communication and Networking - FGNC , 2008, Volume1, pp. 17-20, ISBN: 978-0-7695-3431-2, URL: http://www.cs.virginia.edu/sigbed/archives/2008-10/NetTopo_SIGBEDReview.pdf
- [30] Sangho Yi, Hong Min, Yookun Cho, Jiman Hong, "SensorMaker: A Wireless Sensor Network Simulator for Scalable and Fine-Grained Instrumentation",

- computational science and its application-ICCSA, Volume 5072/2008, pp. 800-810, 2008, URL: <http://www.springerlink.com/content/135t337v633v6240/>
- [31] András Varga, “The OMNeT++ Discrete Event Simulation System”, In the Proceedings of the European Simulation Multi conference (ESM'2001), Prague, Czech Republic, June 6-9, 2001.
- [32] C. Mallanda, A. Suri, V. Kunchakarra, S.S. Iyengar, R. Kannan, and A. Durresi, ”Simulating Wireless Sensor Networks with OMNeT++”.
- [33] Mobility Framework for OMNET++. [Online]. Available: <http://mobility-fw.sourceforge.net>
- [34] Consensus: Collaborative Sensor Networks [Online]. Available: <http://www.consensus.tudelft.nl>
- [35] Sundresh S., Kim W., & Agha G., “SENS: A Sensor, Environment and Network Simulator”, Paper presented at the Proceedings of the 37th annual symposium on Simulation, 2004.
- [36] TrueTime 1.5 - Reference Manual; Department of Automatic Control, Lund University: Lund, Sweden, 2007. Available at: <http://www.control.lth.se/documents/2007/ohl+07tt.pdf> (accessed September 9, 2009)
- [37] Alberto Cardoso, Sérgio Santos, Amâncio Santos and Paulo Gil, “Simulation Platform for Wireless Sensor Networks based on the TrueTime Toolbox”.
- [38] Baldwin, Philip, Sanjeev Kohli, Edward A. Lee, Xiaojun Liu and Yang Zhao, “Modeling of Sensor Nets in Ptolemy II”. In IPSN'04: Proceedings of the Third International Symposium on Information Processing in Sensor Networks., pp. 359–368, ACM Press.2004.
- [39] Opnet Modeler, [online], URL: <http://www.opnet.com/>
- [40] Constantinos Hilas and Anastasios Politis,” Simulations of various IEEE 802.11b network configurations for educational purposes”.
- [41] Prof. Satish K. Shah, Ms. Sonal J Rane, Ms. Dharmistha D Vishwakarma, “Performance Evaluation of Wired and Wireless Local Area Networks”, International Journal of Engineering Research and Development ISSN: 2278-067X, Volume 1, Issue 11, PP.43-48, (July 2012), www.ijerd.com

Chapter 4

- [1] Mohammad Ilyas, and Imad Mahgoub, “Handbook of sensor networks: compact wireless and wired sensing systems”, CRC Press LLC, 2005.

- [2] Roy, P. L.-H. S., “Low-Power Wake-Up Radio for Wireless Sensor Networks”, *Mob. Netw. Appl.*, 15(2), 226-236. doi: 10.1007/s11036-009-0184-3, 2010.
- [3] Martin Leopold, “Sensor Network Motes: Portability & Performance”, Ph.D. dissertation.
- [4] Using MATLAB: User Guide, The Mathworks Inc.
- [5] Website: WWW.MATHWORKS.COM, The Mathworks Inc.
- [6] SIMULINK6: User Guide The Mathworks Inc.
- [7] M. Ohlin, D. Henriksson, and A. Cervin, “TrueTime 1.5 — Reference Manual,” Department of Automatic Control, Lund University, Sweden, 2007. <http://www.control.lth.se/truetime>.
- [8] Andersson, Martin, Dan Henriksson, Anton Cervin and Karl-Erik Årzén, “Simulation of wireless networked control systems”, in *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*. Seville, Spain, 2005.
- [9] T. Chvostek, A. Kratky, M. Foltin,” SIMULATION OF NETWORK USING TRUETIME TOOLBOX”
- [10] TrueTime: “Simulation of Networked and Embedded Control Systems”, March 2006. <http://www.control.lth.se/~dan/truetime/>.
- [11] Martin Andersson, Dan Henriksson, and Anton Cervin. TrueTime 1.3–Reference Manual, Department of Automatic Control, Lund Institute of Technology, 2005.
- [12] D. Henriksson, “TrueTime Simulation of Networked Computer Control Systems”, *Preprints of the 2nd IFAC Conf. on Analysis and Design of Hybrid Systems* (Alghero, Italy), 7-9 June 2006.
- [13] Alberto Cardoso, Sérgio Santos, Amâncio Santo, and Paulo Gil,” Simulation Platform for Wireless Sensor Networks based on the TrueTime Toolbox”.
- [14] K.-E. Arzen, M. Ohlin, A. Cervin, P. Alriksson, D. Henriksson,” Holistic Simulation of Mobile Robot and Sensor Network Applications Using TrueTime”, in *Proceedings of the European Control Conference 2007 Kos, Greece*, pp:4301-08, July 2007,.
- [15] Lund University, TrueTime toolbox, www.control.lth.se/truetime.
- [16] Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma, “A Simulation Study of Behaviour of Wireless Motes With Reference To Parametric Variation” in *International Journal of Advanced Research in Electrical, Electronics*

and Instrumentation Engineering, ISSN 2278 – 8875, Vol. 1, PP:91-95, Issue 2, August 2012.

Chapter 5

- [1] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), October 2003.
- [2] J. Zheng and M. J. Lee. “Will IEEE 802.15.4 Make Ubiquitous Networking a Reality? “: A Discussion on a Potential Low Power, Low Bit Rate Standard. IEEE Communications Magazine, 42(6): 140–146, 2004.
- [3] J. A. Gutierrez, M. Naeve, E. Callaway, V. Mitter, and B. Heile. IEEE 802.15.4, “A Developing Standard for Low-Power Low-Cost Wireless Personal Area Networks”, IEEE Network Magazine, 15(5): 12–19, 2001.
- [4] E. Callaway, P. Gorday, L. Hester, J. A. Gutierrez, M. Naeve, B. Heile, and V. Bahl, “Home Networking with IEEE 802.15.4”, A Developing Standard for Low-Rate Wireless Personal Area Networks. IEEE Communications Magazine, 40(8): 70–77, 2002.
- [5] See <http://www.zigbee.org/>; a brief slide set on ZigBee entitled “ZigBee Overview” can be found under <http://www.zigbee.org/en/resources>.
- [6] Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR- WPANs), IEEE SA Standards Board Std. 802.15.4, 2003.
- [7] Petr Jurčík and Anis Koubâa, “The IEEE 802.15.4 OPNET Simulation Model: Reference Guide v2.0”, Technical Report.
- [8] Crossbow Technology, Inc., Micaz Mote Datasheet. [Online]. Available: <http://www.xbow.com>, 2009.
- [9] Petr Jurcik, “Real-time Communication over Cluster-tree Wireless Sensor Networks”, a dissertation thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (Ph.D.), Department of Control Engineering, Czech Technical University in Prague, January 2010.
- [10] Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma, “Analytical Approach for Performance of Wireless Sensor Networks” in

International Journal of Electronics and Computer Science Engineering ,PP.1877-1884, ISSN- 2277-1956.

Chapter 6

- [1] Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE SA Standards Board Std. 802.15.4, 2006.
- [2] C. P. Singh, O. P. Vyas, and M. K. Tiwari, "A Survey of Simulation in Sensor Networks," in Proc. of the 4th International Conf. on Computational Intelligence for Modelling Control and Automation (CIMCA), pp. 867-872, Dec. 2008.
- [3] K. Fall and K. Varadhan., The Network Simulator 2 (ns-2). [Online]. Available: <http://www.isi.edu/nsnam/ns>
- [4] A. Koubaa, M. Alves, B. Nefzi, and Y. Song, "Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for Time-Critical Events in Wireless Sensor Networks," in Proc. of the 5th Workshop on Real Time Networks (RTN), Jul. 2006.
- [5] A. Varga. (2009) OMNeT++ 4.0 network simulator. [Online]. Available: <http://www.omnetpp.org>
- [6] Petr Jurcik, "Real-time Communication over Cluster-tree Wireless Sensor Networks", a dissertation thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (Ph.D.), Department of Control Engineering, Czech Technical University in Prague, January 2010.
- [7] V. Thirunavukkarasu and M. kannan "Load Density Analysis of Beacon Enabled and Non -Beacon Enabled Wireless sensor Networks", European Journal of Scientific Research, 1450-216X Vol.78 No.1, pp.135-145, 2012.
- [8] Changle Li Huan-Bang Li Kohno, R. State Key, "Performance Evaluation of IEEE 802.15.4 for Wireless Body Area Network", IEEE International Conference on Workshops 2009, pp: 1-5, June 2009.
- [9] Jianliang Z. and M. J. Lee. "A comprehensive performance study of IEEE 802.15.4", Sensor Network Operations, IEEE Press, pp: 1-14, 2004.
- [10] Jianliang Z., Myung J. Lee, "Will IEEE 802.15.4 Make Ubiquitous Networking a Reality?: A Discussion on a Potential Low Power, Low Bit Rate Standard", IEEE Communications Magazine, 42(6), pp: 140-146, June 2004.
- [11] OPNET Technologies, Inc., Opnet Modeler Wireless Suite - ver. 11.5A, <http://www.opnet.com>.
- [12] IEEE 802.15.4 OPNET Simulation Model, <http://www.open-zb.net>

- [13] A. Koubaa, M. Alves, and E. Tovar, "IEEE 802.15.4 for Wireless Sensor Networks: A Technical Overview," IPP-HURRAY Technical Report (TR-050702), Jul. 2005.
- [14] MICAz Datasheet, <http://www.xbow.com>
- [15] Jurcik, P., Koubaa, A., Alves, M., Tovar, E. and Hanzalek, Z, "A simulation model for the IEEE 802.15.4 : delay/throughput evaluation of the GTS mechanism", in 15th IEEE International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems(MASCOTS 2007), October, pp.109–116, Istanbul, Turkey, 2007.
- [16] A. Koubaa, M. Alves, and E. Tovar, "GTS Allocation Analysis in IEEE 802.15.4 for Real Time Wireless Sensor Networks," Workshop on Parallel and Distributed Real Time Systems (WPDRTS'06), Apr. 2006.
- [17] M.N. Hassan and Robert Stewart," Analysis of buffer size dimensioning in GTS enabled IEEE 802.15.4 WSN for real-time applications", Int. J. Mobile Network Design and Innovation, Vol. 3, No. 4, 2011.
- [18] Ms. Sonal J. Rane," Throughput Optimization in IEEE 802.15.4 Using GTS Mechanism" in International Journal of Latest Research in Science and Technology ISSN (Online):2278-5299, Volume 2,Issue 1 :Page No.544-547 , January-February (2013), <http://www.mnkjournals.com/ijlrst.htm>
- [19] Mangharam, R., Rowe, A., Rajkumar, R. and Suzuki, "Voice over sensor networks", Presented at the Real-time Systems Symposium, Rio de Janeiro, Brazil, 2006.
- [20] Melodia, T. and Pudlewski, S., "A case for compressive video streaming in wireless multimedia sensor networks", Focused Technology Advance Series-IEEE COMSOC MMTC, Vol. 4, No. 9, October, available at <http://www.eng.buffalo.edu/~tmelodia/papers/cd-ttmc.pdf>.

Chapter 7

- [1] Zadeh, L. A., "Fuzzy Logic and Soft Computing: Issues, Contentions and Perspectives", in IIZUKA'94: 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing, pages 1-2, Iizuka, Japan, 1994.
- [2] Andrea Bonarini, "Soft Computing Introduction" , Artificial Intelligence and Robotics Lab, Department of Electronics and Information Politecnico di Milano.
- [3] Carlos Gershenson, " Artificial Neural Networks for Beginners", C.Gershenson@sussex.ac.uk

- [4] Rojas R., “Neural Networks: A Systematic Introduction”, Springer, ISBN 3-540-60505-3, Germany, 1996.
- [5] Gurney, K., “An Introduction to Neural Networks”, Routledge, ISBN 1-85728-673-1 London, 1997.
- [6] Andrej Krenker, Janez Bešter and Andrej Kos, “Introduction to the Artificial Neural Networks”. Artificial Neural Networks - Methodological Advances and Biomedical Applications.
- [7] T. Tollenaere, SuperSAB: fast adaptive back propagation with good scaling properties, Neural Networks, page: 561-573, 1990.
- [8] Website: WWW.MATHWORKS.COM , The Mathworks Inc.
- [9] SIMULINK: User Guide The Mathworks Inc.
- [10] Using MATLAB: User’s Guide, The Mathworks Inc.
- [11] Using Neural Network: User Guide, The Mathworks Inc.
- [12] M.N. Hassan and Robert Stewart, “Analysis of buffer size dimensioning in GTS enabled IEEE 802.15.4 WSN for real-time applications”, Int. J. Mobile Network Design and Innovation, Vol. 3, No. 4, 2011.
- [13] Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma,” Soft Computing Technique Based Throughput Optimization of GTS mechanism for IEEE 802.15.4 Standard” in International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, 2013.
- [14] Brown M., Harris C.J., “Neuro fuzzy adaptive modeling and control”, Prentice Hall, 1994.
- [15] MATLAB/GUI User Guide ,www.mathworks.com.
- [16] Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma,” Soft GTS Mechanism Simulator in IEEE 802.15.4 WSN” in International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X ,Volume 3, Issue 9, September 2013.

Chapter 8

- [1] Celebrating 20 years of innovation. Xcell Journal, (48), 2004.
- [2] R. Raeisi and A. Kabir, “Implementation of Artificial Neural Network on FPGA”, American Society for Engineering Education, Indiana and North Central Joint Section Conference, 2006.
- [3] Assist. Prof. Dr. Hanan A. R. Akkar, 2M. Sc. Student Firas R. Mahdi, “Implementation of Digital Circuits Using Neuro - Swarm Based on FPGA “,

- International Journal of Advancements in Computing Technology Volume 2, Number 2, June, 2010.
- [4] BROWN, S.D., FRANCIS, R.J., VRANESIC Z.G, "Field Programmable Gate Arrays", Kluwer Academics Publishers, 1992.
 - [5] STEVENSON, M., WEINTER, R. and WIDOW, B, "Sensitivity of Feedforward Neural Networks to Weigh Errors", IEEE Transactions on Neural Networks, Vol.1, No 2, pp71-80, 1992.
 - [6] BLAKE, J.J., MAGUIRE, L.P., MCGINNITY, T.M., ROCHE, B., MCDAID, L.J, "The Implementation of Fuzzy Systems, Neural Networks using FPGAs", Information Sciences, Vol. 112, pp. 151-168, 1999.
 - [7] KRIPS, M., LAMMERT T., and KUMMERT, A, "FPGA Implementation of a Neural Network for a Real-Time Hand Tracking System", Proceedings of first IEEE International Workshop on Electronic Design, Test and Applications, 2002.
 - [8] HAYKIN, S, "Neural Networks A Comprehensive Foundation", 2nd edition, Prentice Hall Publishing, New Jersey 07458, USA, Vol.1, pp 6-7, 1999.
 - [9] Xilinx Spartan 6 datasheet.
 - [10] Roy, P. L.-H. S., "Low-Power Wake-Up Radio for Wireless Sensor Networks." Mob. Netw. Appl., 15(2), 226-236. doi: 10.1007/s11036-009-0184-3, 2010.
 - [11] Xu, N., Rangwala, S. Chintalapudi, K. K., Ganesan, D., Broad, A., Govindan, R., & Estrin D., "A wireless sensor network for structural monitoring" Paper presented at the Proceedings of the 2nd international conference on embedded networked sensor systems, Baltimore, MD, USA, 2004.
 - [12] Bohli J. M., Hessler A., Ugus O., & Westhoff D., "A secure and resilient WSN roadside architecture for intelligent transport systems" Paper presented at the Proceedings of the first ACM conference on Wireless network security, Alexandria, VA, USA, 2008.
 - [13] Hu W., Bulusu N., Chou C. T., Jha S., Taylor A., & Tran V. N., "Design and evaluation of a hybrid sensor network for cane toad monitoring" ACM Trans. Sen. Netw., 5(1), 1-28. doi: 10.1145/1464420.1464424, 2009.
 - [14] Kulakowski, P., "WIRELESS SENSOR NETWORKS: TECHNOLOGY, PROTOCOLS, AND APPLICATIONS", [Book Review]. IEEE Communications Magazine, 46(6), 42-44, 2008.

- [15] P, R., & P, P. M., “Wireless Sensor Network for Continuous Monitoring a Patient's Physiological Conditions Using ZigBee”, [Article]. Computer & Information Science, 4(5), 104-110. doi: 10.5539/cis.v4n5p104,2011.
- [16] Crossbow Technology Inc, “MTS/MDA Sensor Board User's Manual,” Crossbow Technologies Inc, San Jose, California, 2004.
- [17] Freescale, “MC13201 2.4 GHz Low Power Transceiver for the IEEE® 802.15.4 Standard”, Technical Datasheet, www.freescale.com, 2007.
- [18] Crossbow Technology Inc, “MoteWorks User's Manual,” Crossbow Technologies Inc, San Jose, California, 2004.
- [19] Crossbow Technology Inc, “MoteView User's Manual,” Crossbow Technologies Inc, San Jose, California, 2004.



Appendix A



Development Environments



A.1 Code Generation Using Xilinx ISE 14.6

1. Starting the ISE Design Suite

Follow these steps to launch Project Navigator software and create an ISE project.

1. To start the ISE Design Suite, double-click the Project Navigator icon (Figure A.1) on desktop as shown in Figure , or select **Start > All Programs > Xilinx ISE Design Suite > Xilinx Design Suite > ISE Design Tools > Project Navigator**.



Figure A.1: Project Navigator Desktop Icon

2. Click the New Project button to launch the New Project Wizard or from Project Navigator, select **File > New Project**. The New Project Wizard appears as shown in Figure A.2.

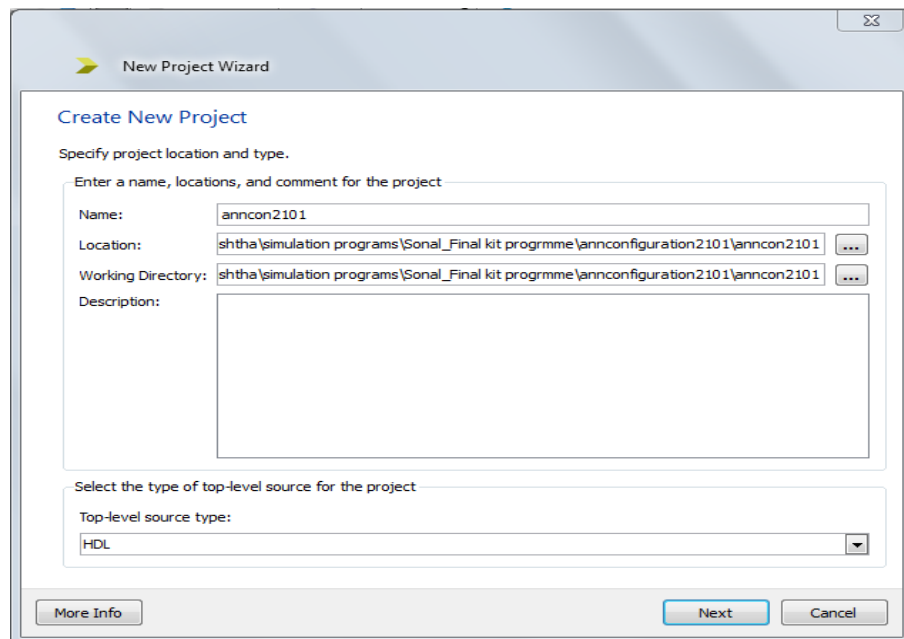


Figure A.2: New Project Wizard—Create New Project Page

3. Verify that HDL is selected as the Top-Level Source Type, and click Next. The New Project Wizard—Device Properties page appears as shown in Figure A.3.
4. In the window, select the device and project properties and change the settings according to FPGA board shown in Figure A.3.

- Click **Next** and then **Finish**, to complete the project creation.

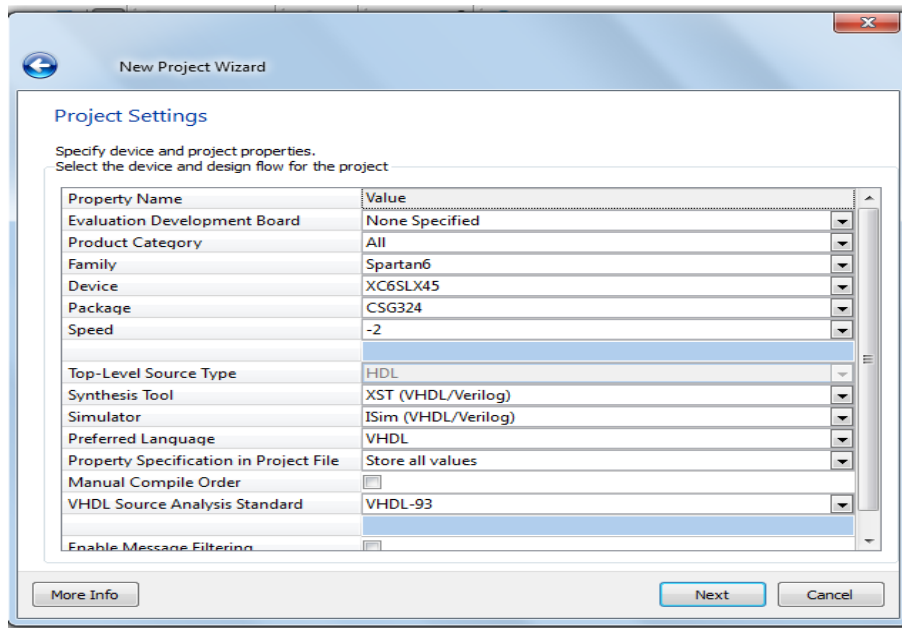


Figure A.3: New Project Wizard—Device Properties Page

2. Adding Source Files to the Project

- Click the Add Source button in the Design Panel toolbar to select the sources provided for tutorial.
- In the next window, make sure that the association and libraries have been properly specified for the tutorial sources. Compare setting with those in Figure A.4.
- Click **OK**.

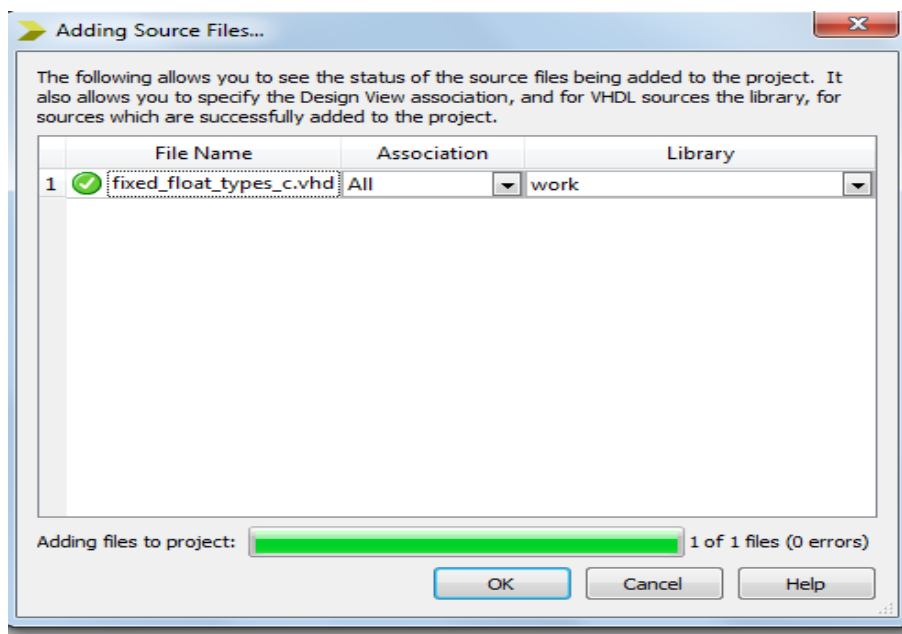


Figure A.4: Adding VHDL Test Bench

3. Launching a Behavioral Simulation

1. Now that the ISE project has been created for the tutorial design, we can proceed to set up and launch a behavioral simulation using ISim.

Setting Behavioral Simulation Properties

- ✘ To set behavioral simulation properties in ISE : In the Design Panel, select **Behavioral Simulation** from the dropdown list.
 - ✘ We should now see the simulation processes available for the design in the Processes pane. (Refer to Figure A.5)
2. Right-click **Simulate Behavioral Model** under the ISim Simulator process and select **Properties**. The ISim Properties dialog box displays (Refer to Figure A.6).
- ✘ In this window we can set different simulation properties, such as simulation runtime, waveform database file location, and even a user-defined simulation command file to launch the simulation.
 - ✘ For the purposes of this tutorial, we will disable the feature that runs the simulation for a specified amount of time.
3. In the ISim Properties dialog box, uncheck the property **Run for Specified Time**, and click **OK**. (Refer to Figure A.6)

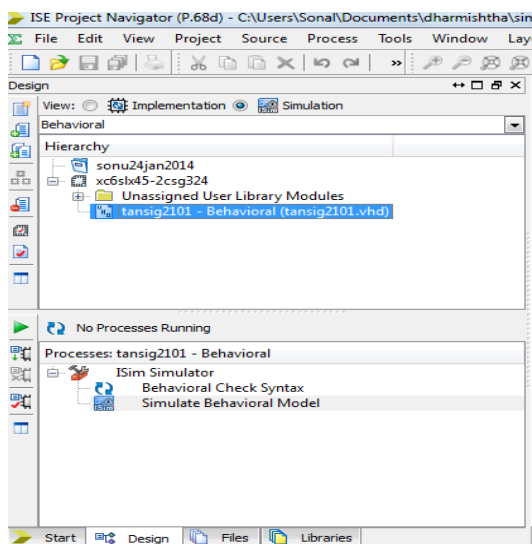


Figure A.5: Process Pane

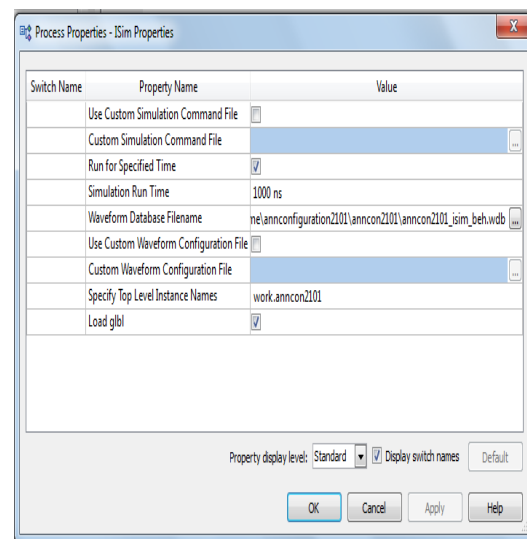


Figure A.6: ISim Properties Dialog Box

Now it is ready to launch the ISE Simulator to perform a behavioral simulation of the tutorial design. To launch the simulator:

In the Processes panel, double-click **Simulate Behavioral Model**. The ISim Graphical User Interface (GUI) (Figure A.7) will appear shortly after the design is successfully parsed and compiled.

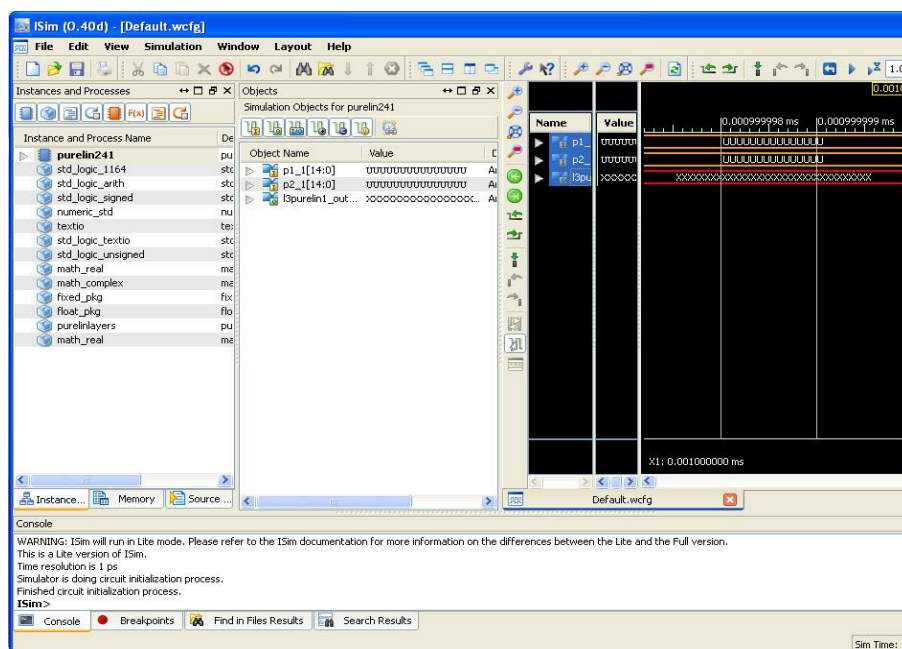


Figure A.7: ISim Graphical User Interface

A.2 Impact Setup

The Digilent Plug-in for Xilinx tools allows Xilinx software tools to directly use the Digilent USB-JTAG FPGA configuration circuitry. For 14.x, Xilinx Impact, ChipScope Pro, EDK Xilinx Microprocessor Debugger (XMD) command line mode, and EDK Xilinx Software Development Kit (SDK) are currently supported by the Plug-in. Refer to <http://www.xilinx.com/> for more information about these Xilinx design tools.

Software Versions Tested:

- ❖ Xilinx ISE Design Suite Version 14.x only (Refer to <http://www.digilentinc.com/> for versions of the plugin for later Xilinx ISE versions)
- ❖ Digilent Adept System 2.9 (or Digilent Adept Runtime 2.9 for Linux) or greater

Microsoft Windows 32-bit and 64-bit Operating Systems

Linux: Red Hat and CentOS 4, 5, 6 (x86/x64), and SUSE 11 (x86/x64)

⌘ Plug-In Installation

To begin, ensure that Xilinx ISE Suite (14.x only) and Digilent Adept System 2.9 (or greater) for Windows, or Digilent Adept Runtime 2.9 (or greater) for Linux, is

installed on the host computer. For Windows Systems also ensure that Microsoft Visual C++ 2008 Service Pack 1 Redistributable Package MFC Security Update is installed on the host computer. The Visual C++ Package is available for download at the following website: <http://www.microsoft.com/en-us/download/details.aspx?id=26368>

The plug-in files **libCseDigilent.dll** and **libCseDigilent.xml** must be copied to a location that is searched by the ISE Suite. Xilinx Impact is used to download bitstreams to FPGA boards. The following steps show how to use Impact with the Plug-in.

1. Launch Impact, double click on “Boundary Scan”, and select the “Cable Setup...” menu item from the “Output” menu.(Refer Figure A.8)

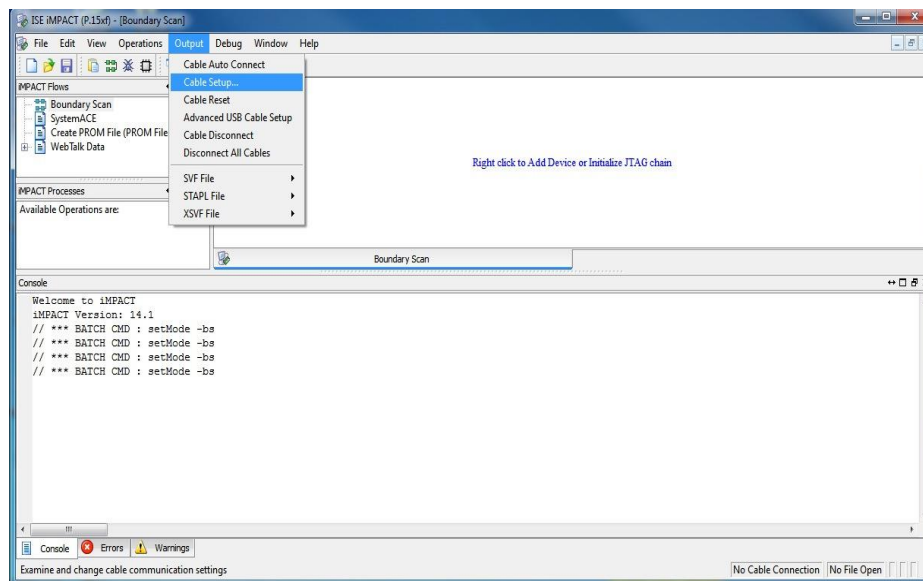


Figure A.8: Cable Set up

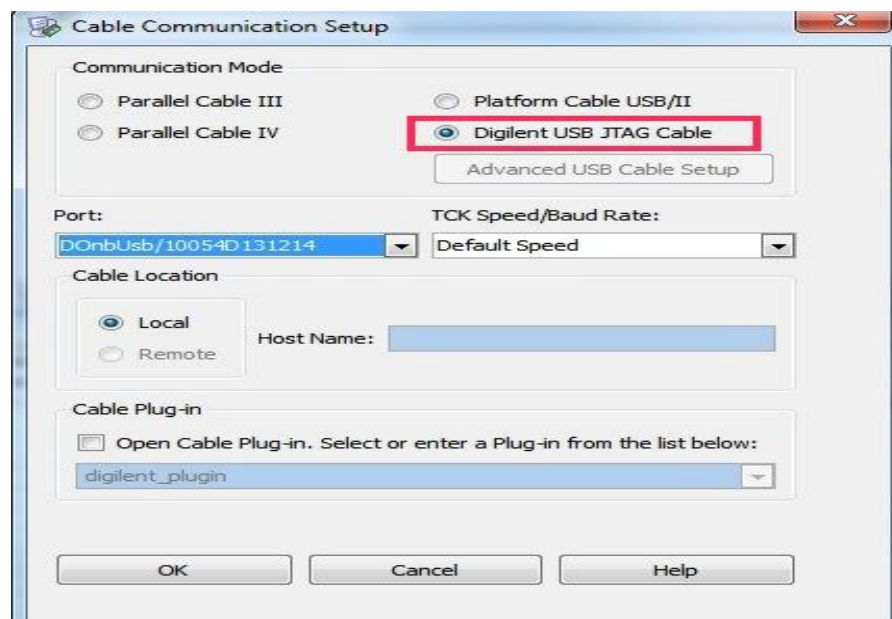


Figure A.9: Cable Communication Setup

2. Select “Digilent USB JTAG Cable” for the “Communication Mode” as shown in Figure A.9.
3. The “Port:” drop-down list should now contain a list of available devices. Select a device to connect to. Click on “OK” and proceed.
4. Right Click in the “Boundary Scan” window to and then click on “Initialize Chain”. (Refer Figure A.10)

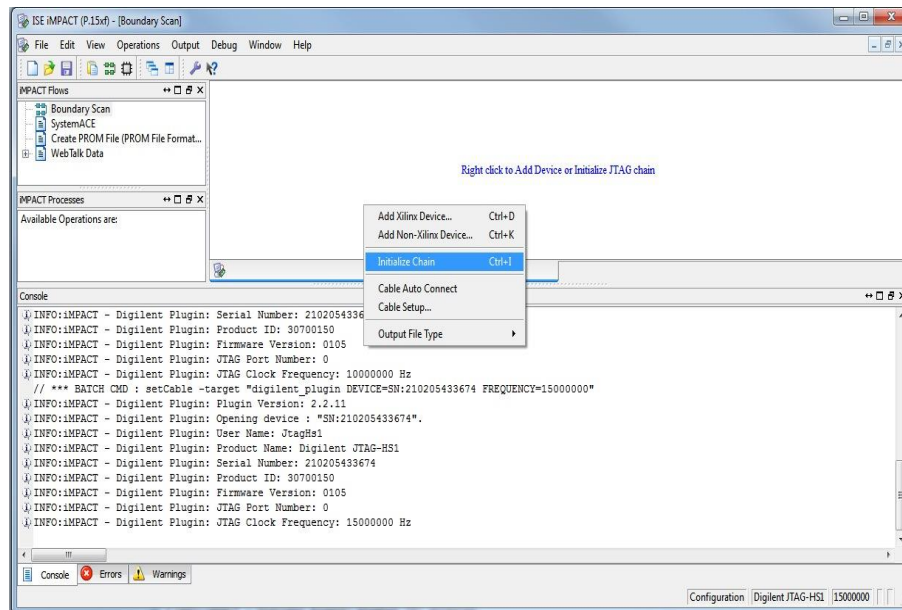


Figure A.10: Snapshot of Boundary Scan

5. Impact is now ready to communicate with the FPGA on the board. Now double click on program option by right click on target device or double click on program in “Impact Processes” Pane, Which will now program the FPGA Chip and display the message as shown in Figure A.11.

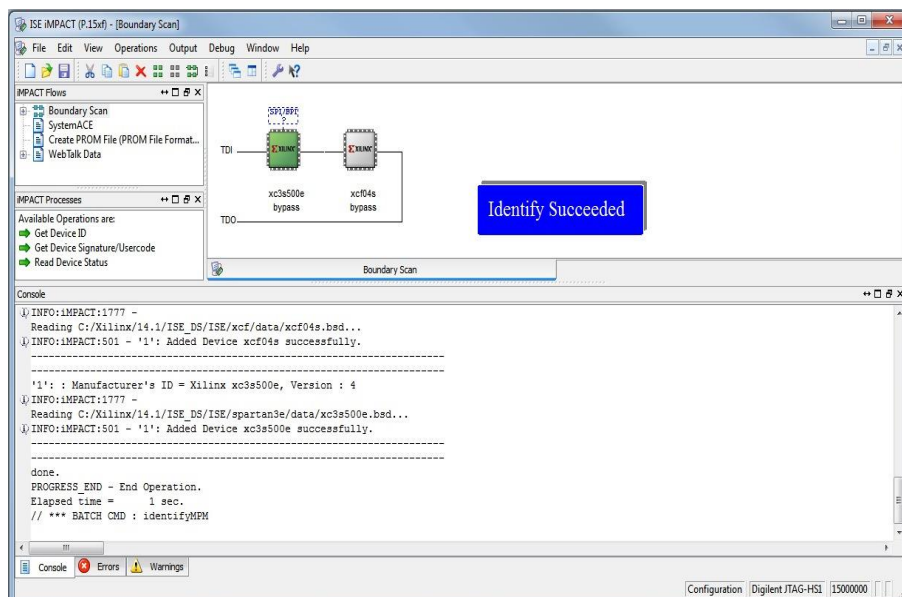


Figure A.11: Program Launch Window

A.3 Atlys™ Board: Spartan-6 XC6SLX45 CSG324C

The Spartan-6 LX45 is optimized for high-performance logic and offers:

- ✘ 6,822 slices, each containing four 6- input LUTs and eight flip-flops
- ✘ 2.1Mbits of fast block RAM
- ✘ four clock tiles (eight DCMs & four PLLs)
- ✘ six phase-locked loops
- ✘ 58 DSP slices
- ✘ 500MHz+ clock speeds

Features:

- ✘ Xilinx Spartan-6 LX45 FPGA, 324-pin BGA package
- ✘ 128Mbyte DDR2 with 16-bit wide data
- ✘ 10/100/1000 Ethernet PHY
- ✘ on-board USB2 ports for programming and data transfer
- ✘ USB-UART and USB-HID port (for mouse/keyboard)
- ✘ two HDMI video input ports and two HDMI output ports
- ✘ AC-97 Codec with line-in, line-out, mic, and headphone
- ✘ real-time power monitors on all power rails
- ✘ 16Mbyte x4 SPI Flash for configuration and data storage
- ✘ 100MHz CMOS oscillator
- ✘ 48 I/O's routed to expansion connectors
- ✘ GPIO includes eight LEDs, six buttons, and eight slide switches
- ✘ ships with a 20W power supply and USB cable

A.4 Programming WSN Notes

1. Programmer's Notepad 2

MoteWorks includes a version of Programmer's Notepad (PN2) that is configured as a simple IDE for nesC code.

- ✘ Steps for implementing WSN application on PN2
1. Open Programmer's Notepad from Start>Programs>Crossbow>PN
 2. Open a nesC file within an application directory. (eg. MyApp.nc), and click on Tools > make MICAz. It can also be executed in the shell commands by clicking on Tools > shell and then typing the command in the dialog box. The "Output" section of the Programmers Notepad will print the compiling results to the screen as shown in Figure A.12.

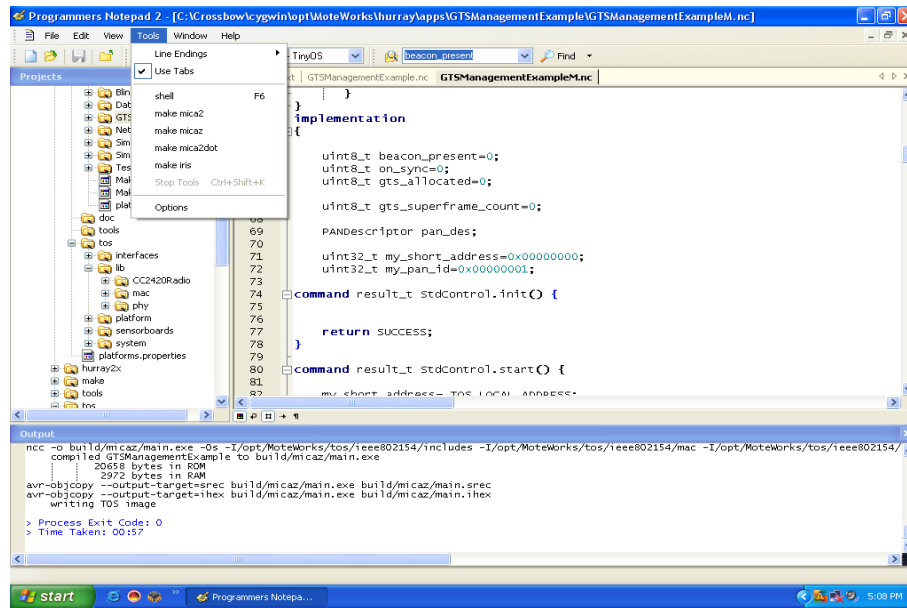


Figure A.12: Snapshot of Programming Notepad 2

After the compilation has completed one should see “writing TOS image” as the last line in the Output window. If we don’t see this then we have made an error typing in one of our files.

3. Application can be installed to a Mote plugged into programming board using Programmer’s Notepad. To install application:

Select Tools > shell. When prompted for parameters, type in **make micaz reinstalls mib520, com10**. The “Output” section of the Programmers Notepad will print the installation results to the screen as shown in Figure A.13.

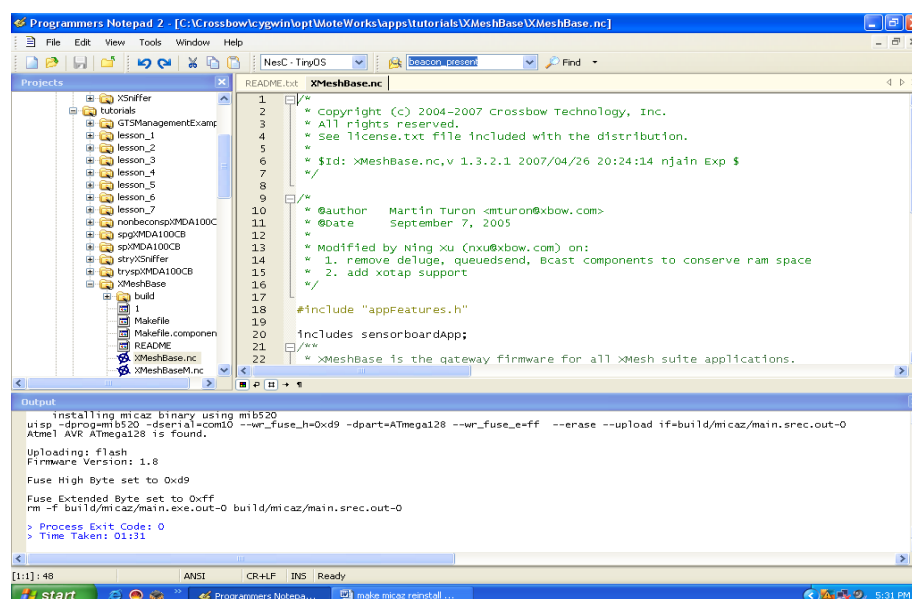


Figure A.13: Output Section of PN2

MoteWorks includes tool named MoteView that can be used to eavesdrop on messages sent over the Mote radios.

2. MOTE-VIEW

MoteView is designed to be an interface between a user and a deployed network of wireless sensors. MoteView provides the tools to simplify deployment and monitoring. It also makes it easy to connect to a database, to analyse, and to graph sensor readings. The key function of the program is to monitor the communications between the gateway and the individual motes. The data can be displayed using the MoteView program.

1. Start MoteView to configure the settings.
2. Start MoteView software, click on “Connect to WSN” icon in the upper left of the panel. The following Figure A.14 is the pop-up window, in the Mode tab, select acquire live data as operation mode.

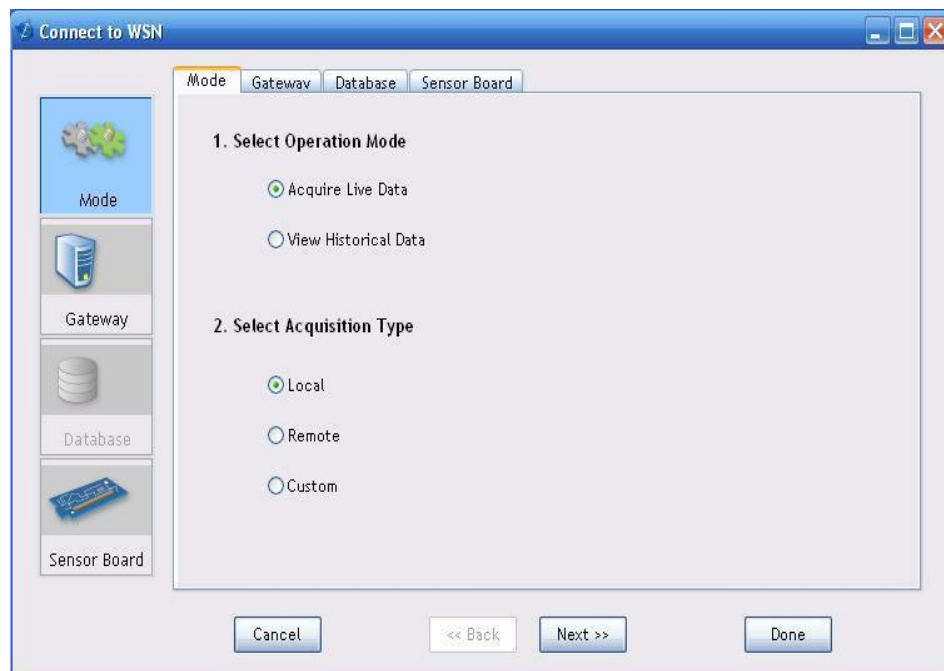


Figure A.14: Mode Configurations

3. In the Gateway tab (Figure A.15), select MIB520 from interface board, COM10 as serial port, and select 57600 as baud rate.

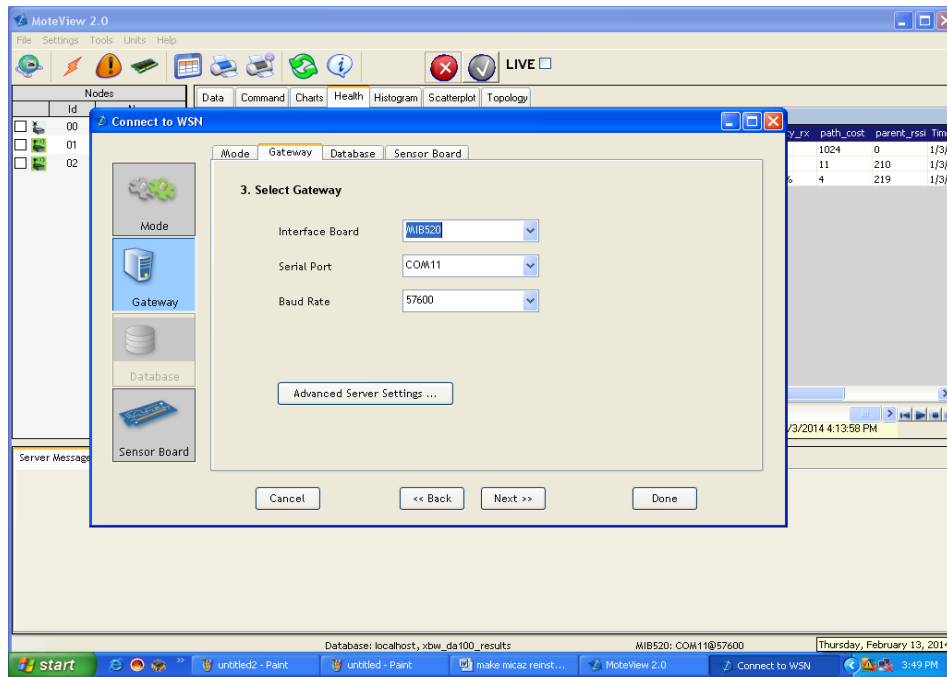


Figure A.15: Gateway Configuration

4. In the Database tab (Figure A.16), localhost is the database server; database name task was created during the PostgreSQL installation.

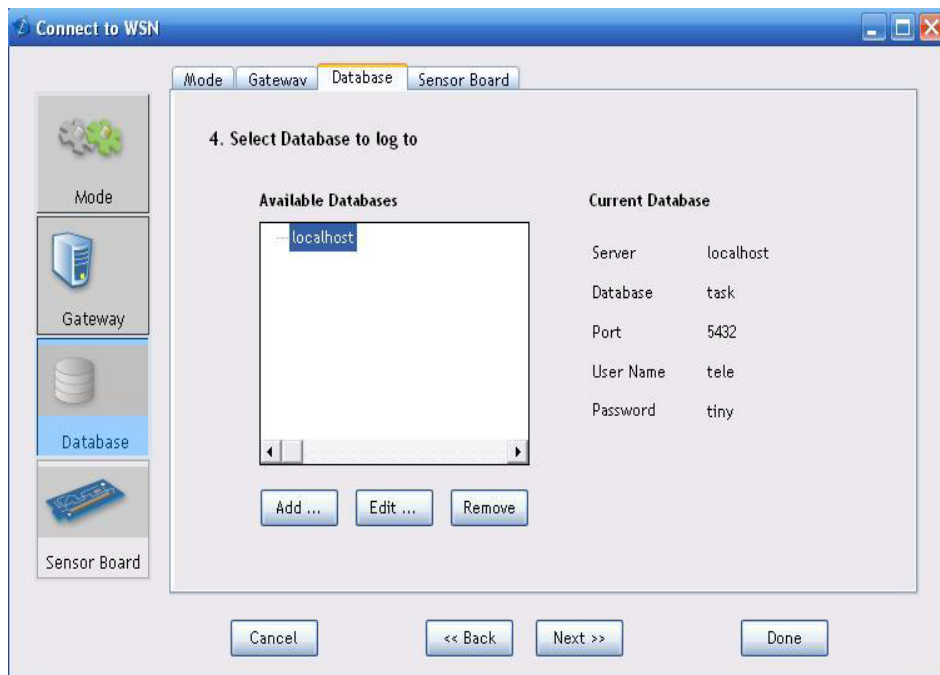


Figure A.16: Database Configurations

5. In the Sensor Board tab (Figure A.17), select XMDA100 from the application name drop list. Then, click "Done". MICAz nodes are flashing and MIB520 base station start to collect data from sensor nodes.

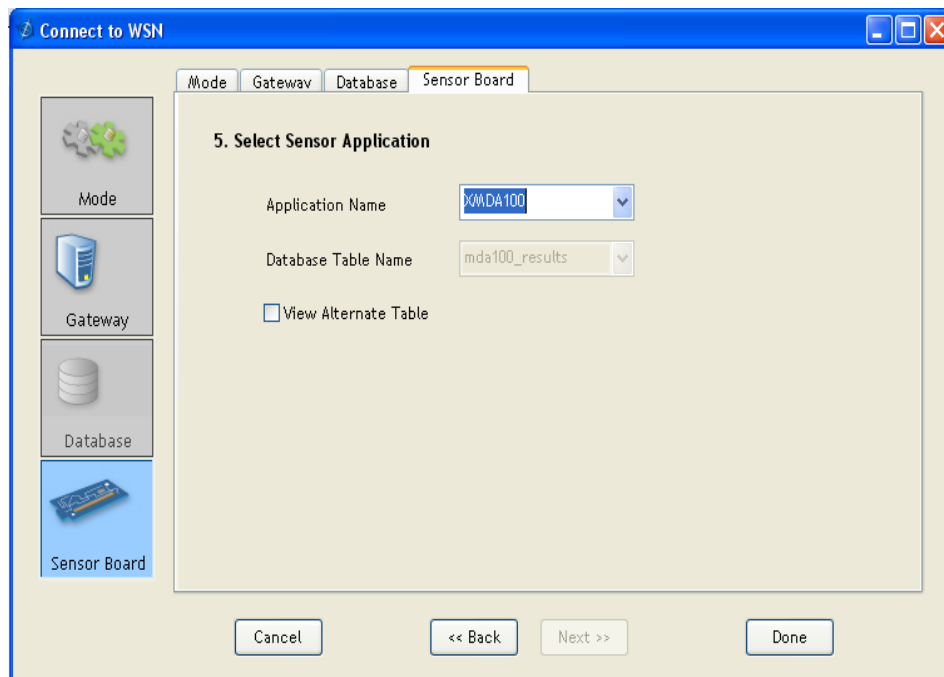


Figure A.17: Sensor Board Configurations

The complete set up for WSN application is shown in following Figure A.18



Figure A.18: MICAz Nodes & Gateway Setup



Appendix B



Photo gallery



Appendix B : Photo gallery

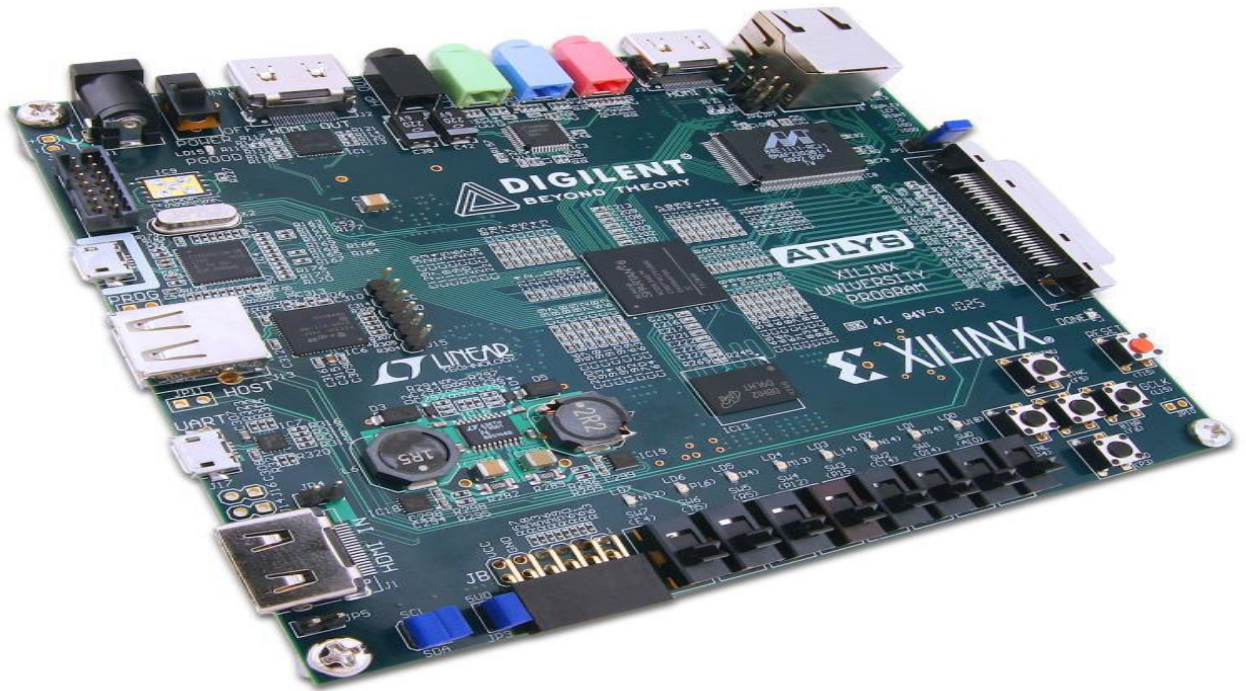


Figure B.1: Spartan-6 XC6SLX45 CSG324C Evaluation Board

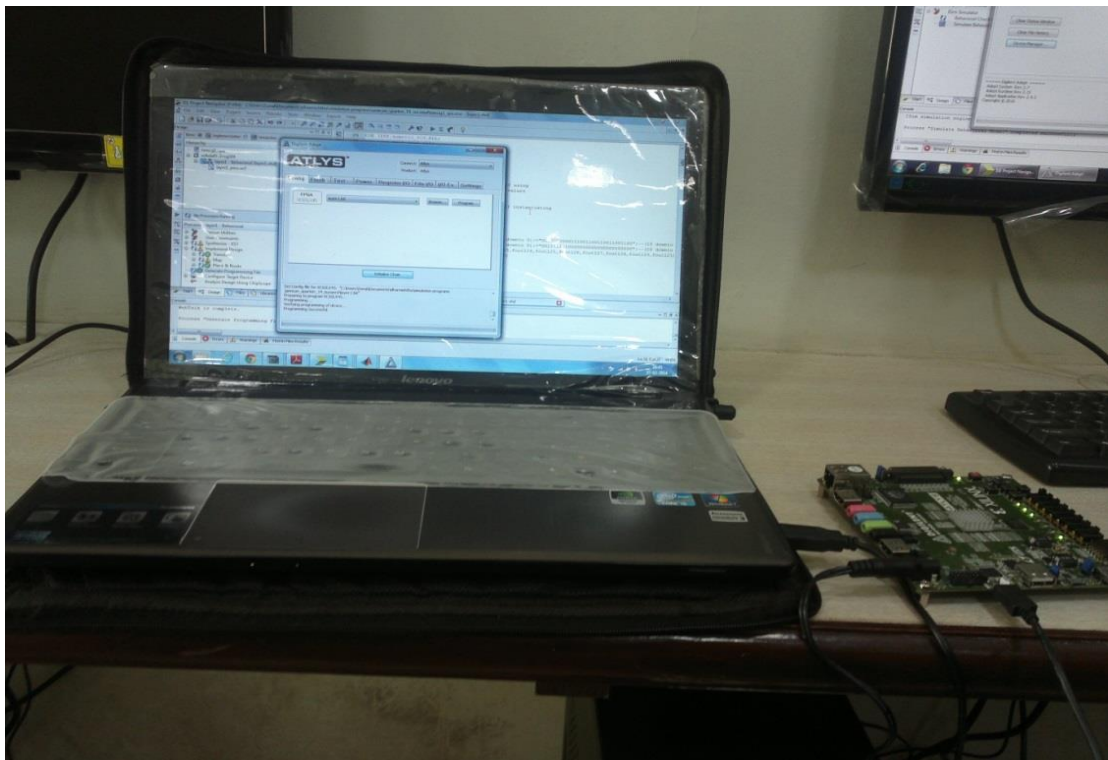


Figure B.2: System Setup for FPGA Implementation of ANN

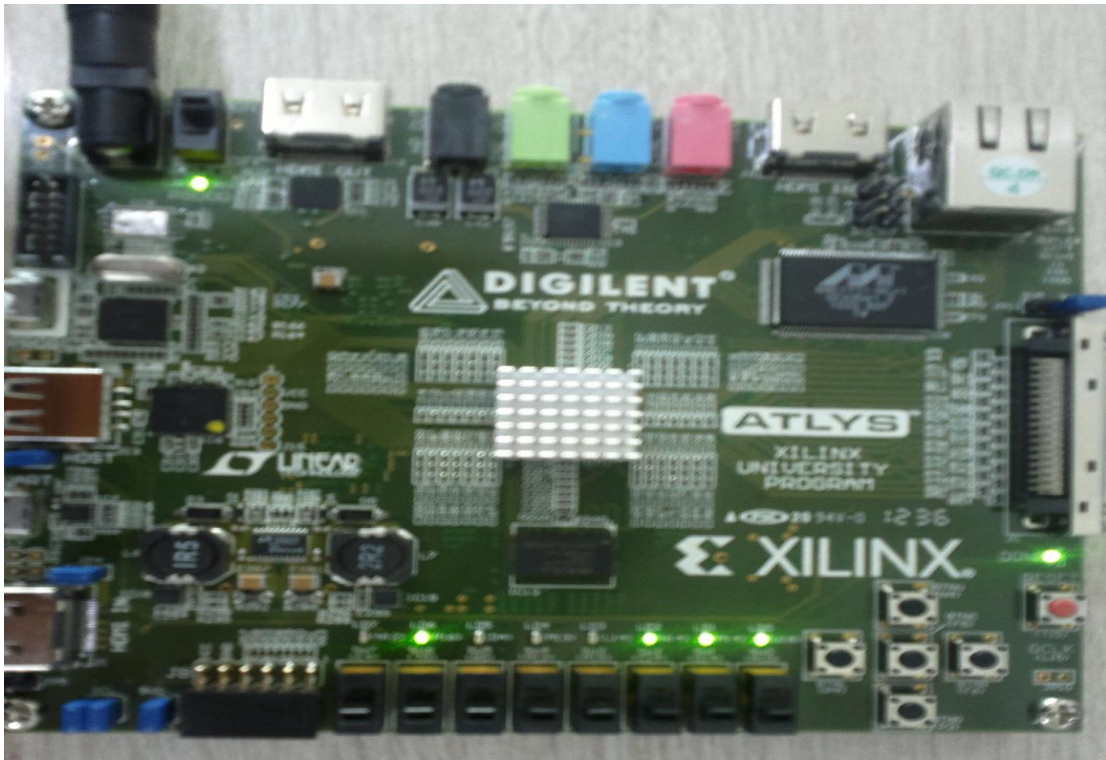


Figure B.3: Output observed on LEDs



Figure B.4: Crossbow's real time Hardware for WSN



Figure B.5: Mote connected on Gateway Setup



Figure B.6: Hardware Setup

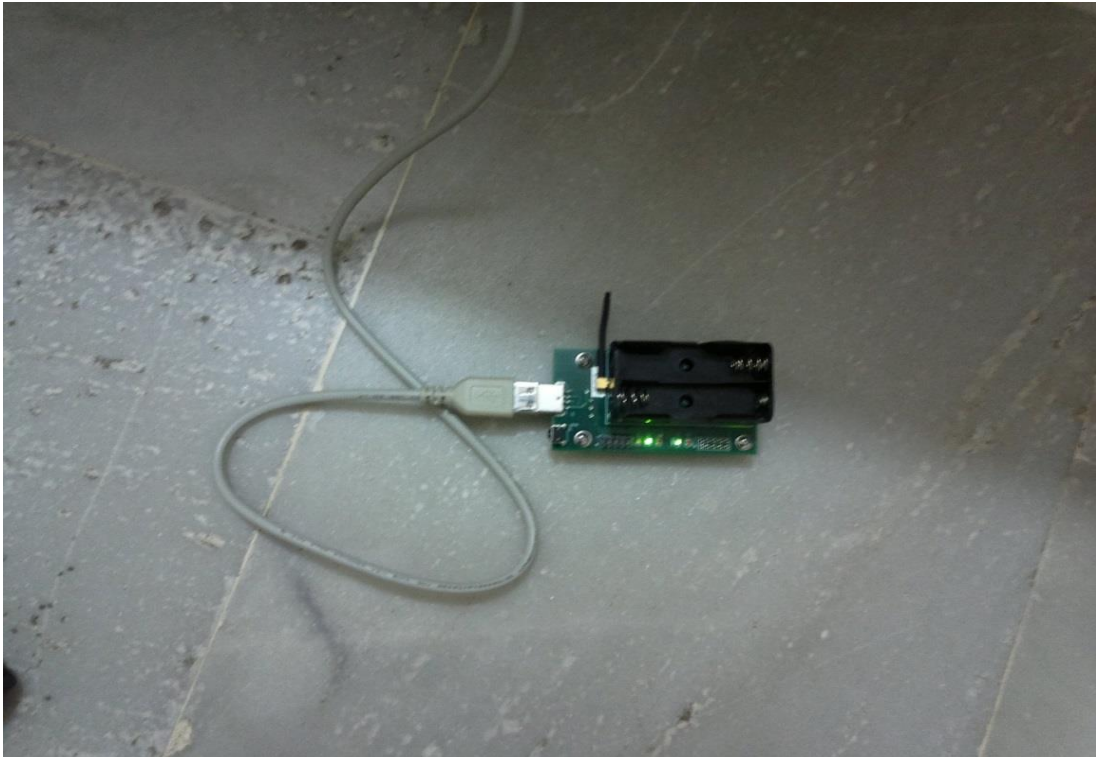


Figure B.7: Status of Mote when data received

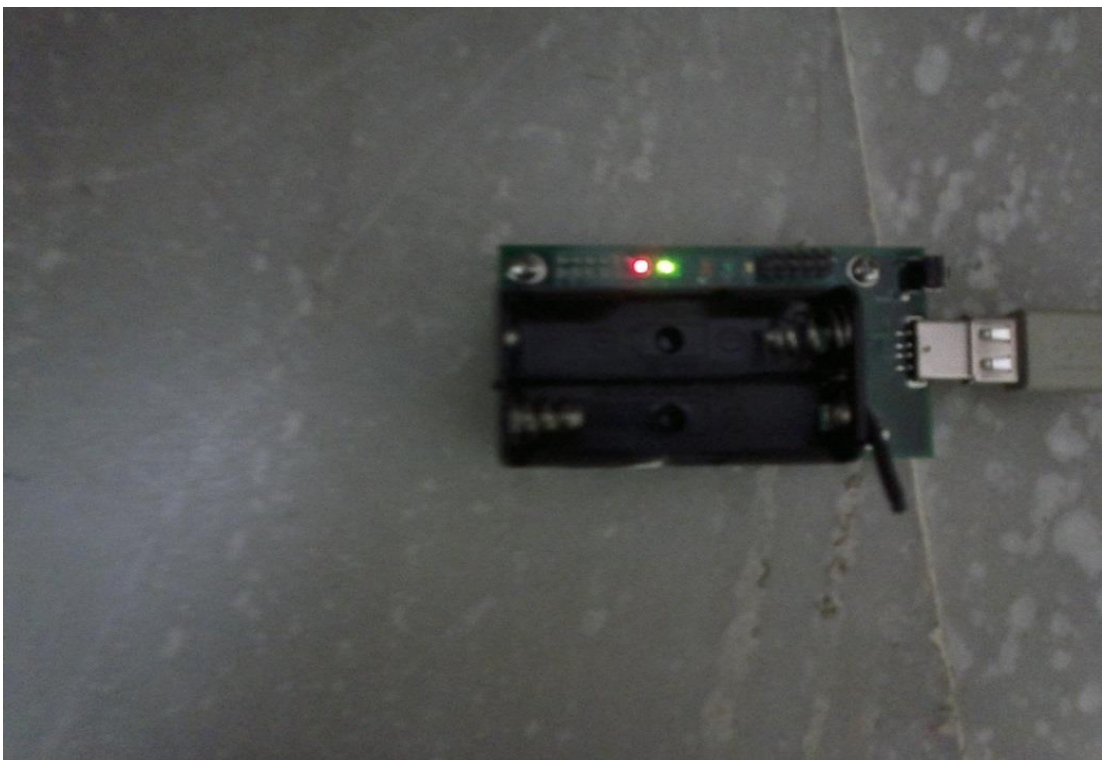


Figure B.8: Snapshot of uploading programme



Appendix C



List of Programmed files & Softwares



Appendix C : List of Programmed files & Softwares

List of Software used to develop the programmes:

Name of software	Company
Matlab and Simulink	Mathworks
Opnet Modeler	OPNET
ISE Design Suit 14.6	Xlinx
MoteWorks_2.0.F	Crossbow
TinyOs	Operating System
Atlys	Digilent
IEEE 802.15.4 OPNET SIMULATION MODEL	OPEN-ZB.net (Open Source)

Matlab and Simulink Files:

Name of the Files	Description
anncon.m	To generate ANN model
anncon.mdl	To evaluate ann model
Anfiscon.m	Gernate anfis model
mote.mdl	Simulink model file to create network using motes of wireless sensor network
Init.m	To initialize the paratmeters of WSN in truetime.
IEEE 802.15.4WSN.fig	Graphical user interface for summary work
IEEE 802.15.4WSN.m	To create links to control GUI parameters
GTS_SIMULATOR.fig	To design simulator using GUI
GTS_SIMULATOR.m	To create links for control parameters

Xilinx Files for ANN Configuration:

Name of the File	Description
Anncon2101.vhd	Vhdl file to write the vhdl code
Layers.vhd	Vhdl package for implementing layers of ANN
Layer_pins.ucf	Assignment of hardware pins of FPGA board to the input and output variables.
Lookuptable.vhd	Lookup table for the Tansig activation function of ann

Moteworks files for WSN hardware :

Name of the File	Description
Makefile	File containing the dependencies (other files) of application uses during the compilation step.
Makefile.component	Describes the top level application component, MoteWorks convention to describe the particular dependencies for a particular application.
Application.nc	The application's configuration
ApplicationM.nc	Application programming code is entered



Appendix D



User Interface



Appendix D: User Interface

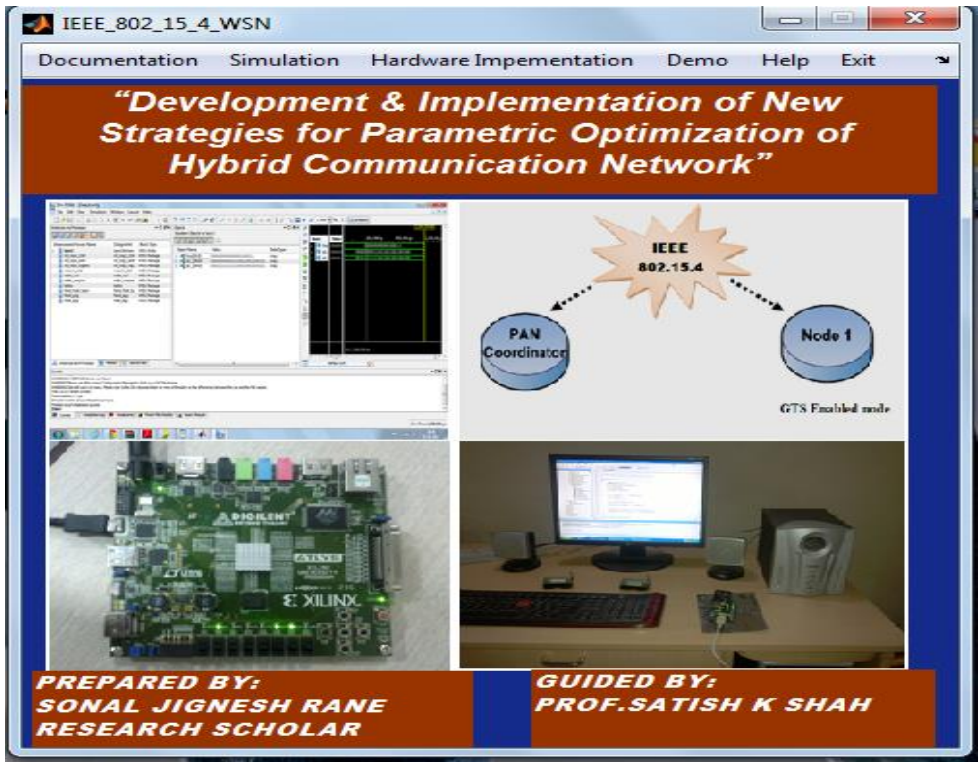


Figure D.1 : IEEE 802.15.4 WSN -User Interface

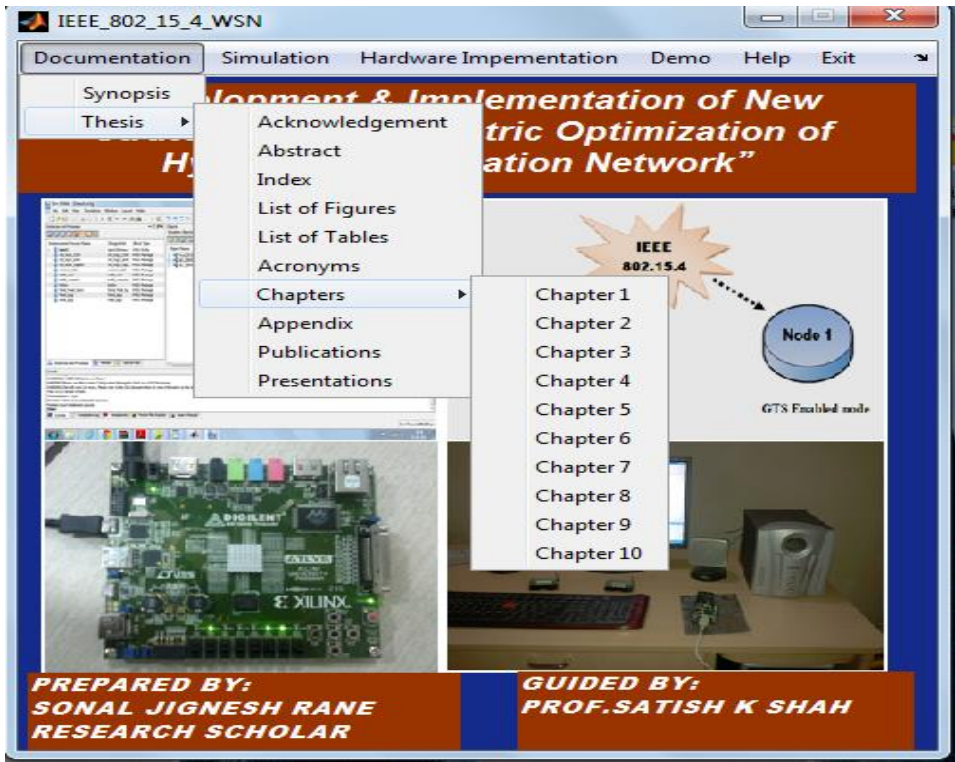


Figure D.2: IEEE 802.15.4 WSN -Documentation

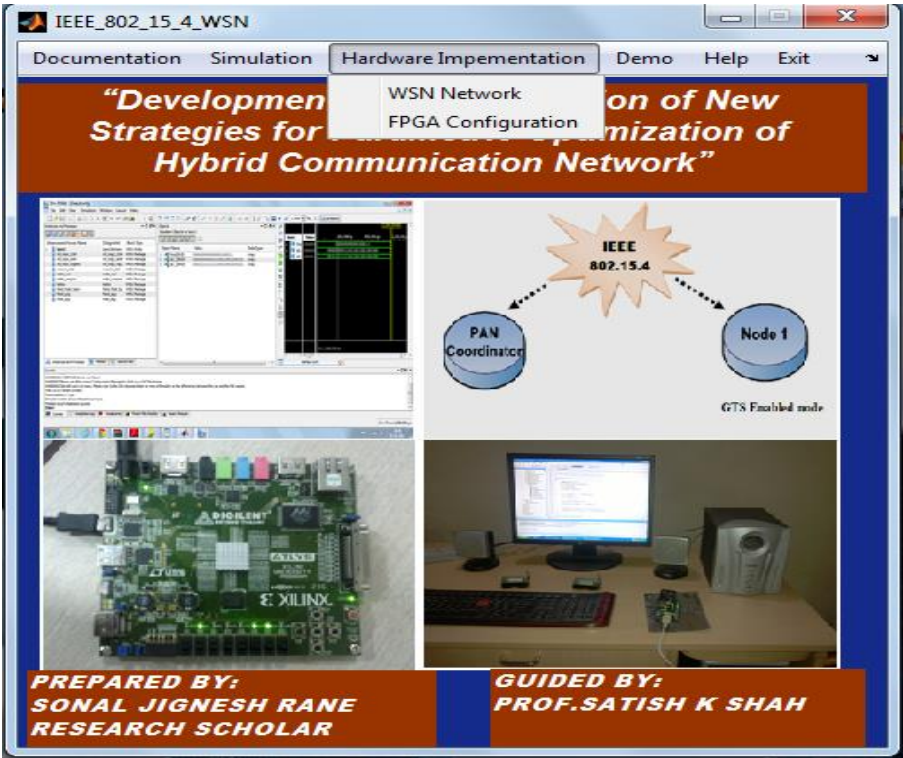


Figure D.3: IEEE 802.15.4 WSN –Hardware Implementation Menu

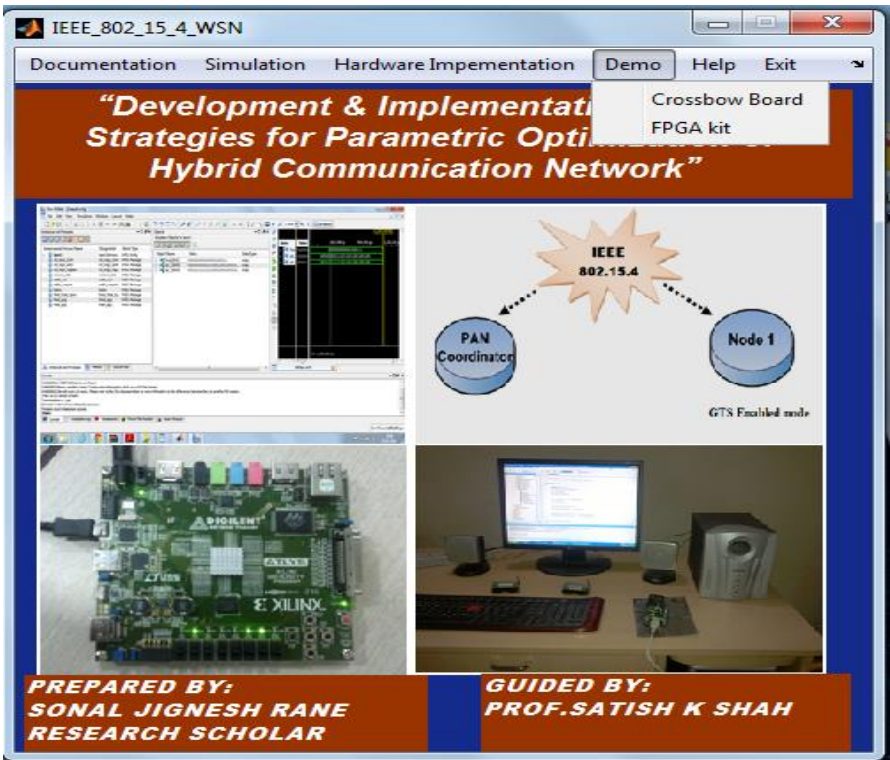


Figure D.4: IEEE 802.15.4 WSN -Demo



Appendix E



**List of Papers
based on**

Research Work



Appendix E: List of Papers based on Research work

Appendix E1:

Publications – Proceedings/Referred (National/International) Journals

1. Ms. Sonal J. Rane, Prof. Satish K. Shah, MS. Dharmistha D Vishwakarma,” Performance Evaluation of Wired and Wireless Local Area Networks” in International Journal of Engineering Research and Development, ISSN: 2278-067X, Volume 1, PP.43-48, Issue 11, July 2012.
2. Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma, “A Simulation Study of Behaviour of Wireless Motes With Reference To Parametric Variation” in International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, ISSN 2278 – 8875, Vol. 1, PP:91-95, Issue 2, August 2012
3. Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma, “Analytical Approach for Performance of Wireless Sensor Networks” in International Journal of Electronics and Computer Science Engineering ,PP.1877-1884, ISSN- 2277-1956.
4. Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma,” Soft Computing Technique Based Throughput Optimization of GTS mechanism for IEEE 802.15.4 Standard” in International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, 2013.
5. Ms. Sonal J. Rane,” Throughput Optimization in IEEE 802.15.4 Using GTS Mechanism” in International Journal of Latest Research in Science and Technology ISSN (Online):2278-5299, Volume 2, Issue 1 :Page No.544-547 , January-February (2013), <http://www.mnkjournals.com/ijlrst.htm>
6. Ms. Sonal J. Rane, Prof. Satish K. Shah, MS. Dharmistha D Vishwakarma,” Soft GTS Mechanism Simulator in IEEE 802.15.4 WSN” in International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X ,Volume 3, Issue 9, September 2013.

Appendix E2:**Presentations-Regional/National/International Conferences**

1. Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma
“SIMULATION STUDY OF BEHAVIOUR OF WIRELESS MOTES WITH
REFERENCE TO PARAMETRIC VARIATION” at a State Level Paper Contest
called Control, Microcomputer, Electronics and Communication
(CMEC_2011)), on Sunday, February 20, 2011 Organized jointly by IETE-
Vadodara Centre and Electrical Engineering Department, Faculty of Technology
& Engineering, M.S.University of Baroda, Vadodara.
2. Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma
“ANALYTICAL APPROACH FOR PERFORMANCE OF WIRELESS SENSOR
NETWORK” at a State Level Paper Contest called “IT Security with 4G
Communication (ITS4GC_2012)”, on Sunday, April 22, 2012 Organized jointly
by IETE-Vadodara Centre and IE(I)-Vadodara Local Centre, Vadodara.
3. Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma,
“Soft Computing Technique Based Throughput Optimization of GTS mechanism
for IEEE 802.15.4 Standard” in International Conference on Soft Computing and
Software Engineering 2013 (SCSE'13), in San Francisco, California, USA.
4. Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma,
“ANN Based Performance Optimization Of GTS Mechanism For IEEE
802.15.4” at a State Level Paper Contest called “Soft Computing for Processing,
Security, Networking and Communication (SCPSNC_2013)”, Organized jointly
by IETE-Vadodara Center and Electrical Engineering Department, Faculty of
Technology & Engineering, M.S.University of Baroda, Vadodara on January 20,
2013.
5. Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma,
“Optimization of IEEE 802.15.4 Parameters” at a State Level Paper Contest
called “Target Technologies in Computing, Automation and Communication
(TTCAC_2014)” on Sunday, March 23, 2014 organized jointly by IETE-
Vadodara Centre and The Institution of Engineers (I)-Vadodara Local Centre,
Vadodara.

Appendix E3: Awards/Prizes

- ✂ First prize in the National Paper Contest called “Soft Computing for Processing, Security, Networking and Communication SCPSNC-2013” held at Faculty of Technology & Engineering, MSU BARODA on 20th January, 2013 organized by IETE Vadodara and EED, FTE, M.S.University of Baroda, Vadodara. Paper entitled “ANN Based Performance Optimization of GTS Mechanism for IEEE 802.15.4 Standard”.