A Reliability Evaluation of Wireless Sensor Network Simulator: Simulation vs. Testbed

Name: ZhengYi Guan (Alex)

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Computing

UNITEC New Zealand, 2011

Abstract

Wireless Sensor Network (WSN) is the new generation of sensor network which combines micro-electronic technology, embedded computing technology, distributed information processing technology, networking and wireless communication technologies. Wireless sensor network is composed of a huge number of sensor nodes that can cooperate to sense, collect and process the data. It has a series of features, such as the huge number of nodes, limited node hardware resource, power supply constraints, dynamic topology and self-organization, etc. The character of WSN allows it to be widely utilized in military surveillance, environmental observation and forecast, household appliance automation, medical care, space exploration, and other various commercial applications. Thus, these all contribute to a broad perspective of wireless sensor network. However, it also has brought a lot of challenges in the application areas – how to test and validate algorithms and protocols of wireless sensor network towards the large scale scenarios. Comparing simulators with test-bed, the accuracy and reliability of simulators becomes a strong issue.

The purpose of this thesis is to present an effort took to examine the reliability of Castalia/OMNET++ simulator through simulating wireless sensor network. In this thesis, two test-bed experiments were setup by using two IRIS (Integrated Resource Information System) nodes and one MIB520 gateway to evaluate the simulator's reliability. The results collected from IRIS test-bed experiments were compared with the results collected from the simulation experiments of the same scenarios on Castalia simulator. A lot of various parameters (traffic load, radio transmission power, and interference model) were considered in the simulation. The simulation results show that the Castalia simulator provide quite reliable results up to 2% over/under estimation of test-bed results. In addition, a simple tuning Castalia default value configurations can significantly increase accuracy.

Acknowledgement

The success of the thesis is a result of the valuable contributions of many people who merit to be mentioned.

Firstly, I would like to express my gratitude and thanks to my principal supervisor Kathiravelu Ganeshan who he has given me a lot of comments, intellectual guidance, advice, and help on my thesis, especially when I was confused.

Secondly, I wish to express my sincere thanks to the Head of Unitec Computing Department –Hossein Sarrafzadeh for giving me advice and financial support on my research topic selection and approving the purchase of experiment hardware.

Thirdly, I would like to give my heartfelt thanks to my wife, she has been very supportive of my study, and she has given me every possible help in living and study during my master study period. I am very grateful to my parents for their infinite love and care, and all my family members for their invaluable contributions to me.

Fourthly, I would also like to appreciate Dr. Guan Yue Hong and Dr. Aaron Chen for their constructive comments and advices.

At last, but not least huge thanks goes to all the people who have given me guidance, help, encouragement and support. I wish you all the best!

Declaration

I ensure hereby declare that this paper is entirely based on my own work and under the guidance of Kathiravelu Ganeshan has not been submitted in whole or in part to any other university.

Signature: _____ Date:_____

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACK	Acknowledgment
ADC	Analog to digital converter
BAN	Body Area Network
CCA	Clear Channel Assessment
CCI	Chip Correlation Indicator
FFD	Full function device
GPS	Global Positioning System
I/O	Input/Output
IRIS	Integrated Resource Information System
LED	Light Emitting Diode
LLC	Logical Link Control
MAC	Medium Access Control
OSI	Open Systems Interconnection
OUI	Organizationally Unique Identifier
PAN	Personal Area Network
PDA	Personal Digital Assistant
PIR	Passive Infrared Sensors
PHY	Physical layer
PRR	Packet Reception Rate
QoS	Quality of Service
RAM	Random Access Memory
RFD	Reduced Function Device
RFID	Radio Frequency Identification
ROM	Read Only Memory
RSSI	Received Signal Strength Indicator
USB	Universal Serial Bus
WSN	Wireless Sensor Network

CHAPTER 1. INTRODUCTION

1.1 Background

Individuals and businesses are continuing to reap many benefits from the rapid developments in computer science and technologies. Wireless sensor networks (WSN) is one area of development in computing that is currently attracting many practical usages. The wireless sensor networks technology combines sensing, communication, power and computation into an integrated small single device. These devices form a broad range of connectivity that even have being utilized extensively around the world.

WSN is a significant technology for the future. If the keyboard, mouse and touch screen have changed the way in which people communicate with computers, automatic control system has changed the interaction method between machines**,** and the Internet has changed the way people communicates with each other, then the wireless sensor network technology is going to change the interaction method between human and nature. From this point of view, the emergence of WSN technology has a significant impact on human beings. According to American 'BusinessWeek' magazine in 1999, the wireless sensor network technology has been predicted to be the one of the most important technologies in the 21st century [5].

The main purpose of wireless sensor networks is to observe an area including detecting, identifying, tracking and localizing one or more items of attention to collect data, and then sending back the collected data by using the wireless transmission mode [1, 2, 3]. For instance, a chemical plant can be easily observed for any chemical leaks. It is completed by hundreds or thousands of wireless sensors which automatically form a wireless sensor network and

immediately report to the local monitoring base station or remote monitoring facility if any chemical leak is detected.

In the daily life, there are thousand applications of sensor networks (wired and wireless) all around us. For example, in Unitec, we have fire sensors, temperature sensors, surveillance cameras and even light switches controlled by Passive Infrared Sensors (PIR) sensors.

Nowadays, evaluating the performance of wireless sensor networks (WSN) only through the experiment is impossible, especially in a wireless sensor network which contains a large number of motes that is very difficult to get through the implementation [4].Therefore, the application of WSN simulators to simulate those complicated experiments has been gradually becoming a possible and necessary solution [4, 43, 57].

As the purpose of this research aims to evaluate the reliability of the collected data via simulating WSN, this study was conducted in a controlled test environment to collect data. The collected data was then analysed and compared with the data collected via simulation.

1.2 Motivation

Compared to a WSN implementing on a test-bed, simulating WSN is a favorable and the most ideal approach for this type of research [57]. Implementing WSN on a real test environment tends to attract other non-favourable factors such as, financial investment, time-consuming, difficulty of implementation, and hardware failure, etc [14, 43]. In contrast, WSN simulators have a number of advantages, for instance, low cost, flexible, efficient, realistic, and easy to setup. In addition, Wireless sensor network simulation technology is the foundation of WSN routing

and MAC protocols research, and WSN application development [41]. As a result, a lot of WSN simulators have been developed, but their reliability and accuracy are still in question and very little research is conducted in this area [14, 43]. To the best of my knowledge, this is the first study conducted in order to evaluate the reliability of Castalia simulator.

1.3 Research Objectives

The research objectives of this thesis are to:

1. Validate the reliability of wireless sensor networks simulations.

2. Research and analyze the most popular simulation packages available for simulating wireless sensor networks.

3. Find out the parameters that can be varied on these packages

4. Design and implement a simple WSN experiment by using two nodes and a base station

5. Design the simulations can be done using the researched simulation

In order to examine the reliability of the WSN simulation, the Crossbow IRIS test-bed will be employed and the results will then be compared with the results of WSN simulation experiments. In addition, different parameters will be set in the WSN simulations and test-bed experiments in order to see their differences.

1.4 Contributions

The contributions of this thesis are to:

1. Present a first effort to evaluate the reliability of Castalia simulator.

2. Consider a very simple test-bed setup made of two IRIS sensor nodes and a MIB520 gateway. Test on it the performance of packet successful rate, packet dropped rate, packet retry rate and energy consumption.

3. Validate the Castalia realistic wireless channel, radio and MAC modules and evaluate the performance of Castalia simulator.

4. The results of the test-bed are compared with the results of the simulations of the same scenarios on Castalia. The experiment design principle considers a simple experimental setup in order to have a clear and manageable environment in which the behaviour of the network is not affected by complex technical issues.

1.5 Organization of Thesis

This paper is organized as follows:
Chapter 1 contains introductory material which introduces the concept, characteristics, application areas and the technologies of wireless sensor network and other related problems.

Chapter 2 presents the characteristics of various WSN simulators with a description of the current popular simulators, and their features. The OMNet++ and Castalia simulators will be interpreted specifically in this chapter, including their framework, structure and language, etc. This chapter will also provide an interpretation regards to how to setup and configure the simulation environment

as well. At last, it presents the related work with this thesis study

Chapter 3 describes the differences in wireless modules and test-bed hardware.

Chapter 4 explains what the experiment design principle and experiment scenarios are. This chapter also deals with the practical issues like, setting up the test-bed and implementing of the experiment.

Chapter 5 explains how the simulations have been setup and also describes the simulation scenarios, such as changing the initial parameters, setting various MAC protocols, MAC frame sizes, transmission power, and packet rates, etc. Propose to consider the interference channel transmission of data collection and conduct simulation experiments.

Chapter 6 compares the simulation and real test-bed data, presents the analysed results, and explains the parameter variations.

Chapter 7 concludes this thesis, describes the suffered difficulties when researching this thesis, and generates a number of future works needed to be researched.

CHAPTER 2. WSN SIMULATORS

This chapter is divided into three sub sections – Overview of WSN, Simulation platform for WSN and Related Work. The objective of Overview of WSN sub chapter is to briefly introduce Wireless Sensor Network (WSN), including its architecture, technologies, wireless communication standards, application areas and the tinyOS operating system /nesC programming language. The aim of Simulation platforms for WSN is to describe the characteristics of the WSN simulation, briefly describe the current popular WSN simulation tools, and detail the OMNET++ and Castalia simulators and their modules.

2.1 Overview of WSN

2.1.1 The Architecture of Wireless Sensor Networks

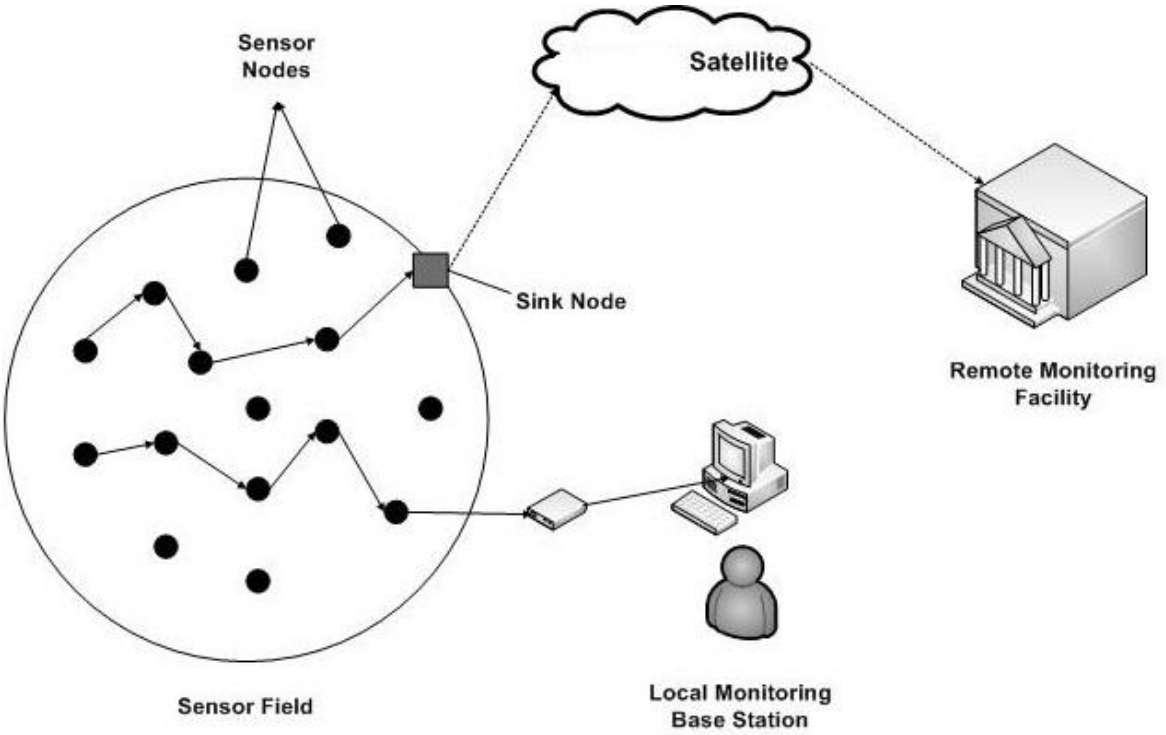

Figure 2-1: The Architecture of Wireless Sensor Network [2]

Wireless sensor network (WSN) [1, 2, 3, 6, 7, 8, 25, 26] is an emerging technology consisted of a large number of low power, low cost, light weight, and small size sensing devices, which is also called motes or nodes, distributed spatially over the physical environments. These nodes are able to form a self-organised network. The specific wireless sensor network structure is presented in Figure 2-1. In the sensor field, sensor nodes collect and transmit sensed data back to sink node (it is also known as Base Station, Gateway, and Access Point) through wireless communication, then sink node will finally forward the sensed data to the users via other communication links (e.g. Satellite) [1, 3, 7]. Sink node can be from a variety of devices, such as portable computer, PDA (Personal Digital Assistant), or ground station, etc

The size of a wireless sensor nodes are usually varied from a shoe box size to the size of a gold coin [2]. The following figure 2-2 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 [6].

Figure 2-2: Typical wireless sensor nodes size [2]

The traditional wireless sensor 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 [2, 3, 7].

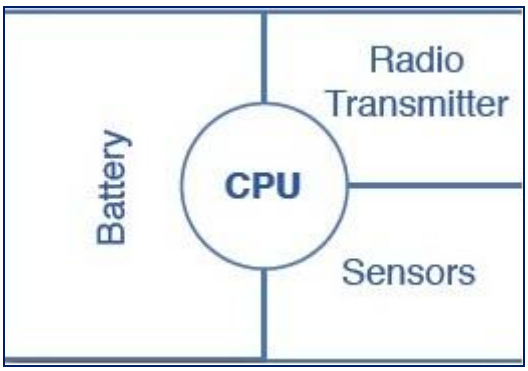

Figure 2-3: Wireless sensor node components [6]

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 [1, 2, 3, 8].

2.1.2 The Characteristics of Wireless Sensor Network

The traditional wireless networks, such as mobile ad hoc networks (MANETs) [58], cellular network [59], and Bluetooth [60] are the short-range wireless local area network [6]. They work mainly through the dynamic routing and mobility management technology to achieve the purpose of multimedia data transmissions, for example, voice, images and video [8]. The primary goal of traditional wireless networks is to provide high quality of service (QoS) and high bandwidth utilisations [7, 25].

Wireless sensor network is the integration of monitoring, control and wireless communication system. Compared with the traditional wireless network, wireless sensor network has many similarities as well as several significant differences,

such as the huge number of nodes, limited node hardware resource, power supply constraints, dynamic topology and self-organization, etc [2, 3, 44]. The primary goal of wireless sensor network is to utilize the energy efficiently, which is also the most important difference between the traditional wireless network and WSN. The following are some of the main characteristics and restrictions of wireless sensor network [1, 2, 6, 7, 8, 25]:

- Very small devices
- Very large deployment scale
- Low cost products
- Energy efficient function
- Self-organization of the network model
- Fault tolerance
- Data-centric network
- Area coverage
- Less human interaction
- Mobility, reliability, flexibility and universality
- Limited node hardware resource
- Limited processing capability
- Limited power supply capability
- Low-rate of data transmission

2.1.3 The Application of Wireless Sensor Networks

Similar to the computer network technology, wireless sensor network was originally proposed by the U.S. military for application in the battlefield environment monitoring [6]. In civilian areas, wireless sensor networks also have a wide range of applications, especially in medical care, environmental monitoring and forecasting, construction management, intelligent transport system [3], space exploration and other fields. The application areas of wireless sensor networks

seem to be in a broad range, which can be summarized into two main categories, military and civilian areas [2].

Military

As wireless sensor networks are composed of intensive, low-cost, random distribution of nodes [1, 2, 3, 25]. In addition, WSN has the self-organization and fault tolerance capability, so that the entire WSN system will not collapse even if some of the nodes were damaged by malicious attack [44]. It is such a unique advantage of the application of WSN in this field which the traditional Network technology cannot match. As a result, the wireless sensor network is very suitable for a harsh battlefield environment [6, 8].

Civilian Areas

In civilian areas, the application of wireless sensor network is indispensable. It can be used to monitor and forecast physical structure health [6] or environmental conditions [10], animal monitoring [7], to observe soil, and climate conditions [11] in agriculture, as well as to monitor the patient's physical health in medicine [8, 27], to monitor and control various household appliances working conditions which we call smart homes [61], to monitor space exploration and hazardous areas of testing and inspection [62], to provide the intelligent transport system which can prevent accidents and post-accident investigations [3] and so on. Figure 2-4 presents some of the most popular application areas of WSNs.

Figure 2-4: WSN application areas [6]

2.1.4 Wireless Sensor Network Technologies

2.1.4.1 Communication Standards

IEEE 802.15.4

IEEE 802.15.4 [8, 9, 22, 23, 24] standard describes the low rate wireless personal area networks (LR-WPANs) in the physical (PHY) layer and the media access control (MAC) layer, so IEEE 802.15.4 protocol is known as Low Rate Wireless Personal Area Networks (LR-WPANs). It belongs to the IEEE 802.15 workgroup released in 2003. The IEEE 802.15.4 standard is the regulation basis of ZigBee [12], WirelessHART [45], and MiWi [46] wireless protocols.

IEEE 802.15.4 protocol defines two types of equipments: Reduced Function Device (RFD) and Full Function Device (FFD) [8, 9, 24]. Wireless personal area network (WPAN) is composed of several RDFs, FFDs, or both of them [9]. In the wireless personal area network, each device exchanges data in accordance with WPAN communication protocol. In a PAN, there is at least one FFD as PAN coordinator and each PAN will have a unique ID [22].

IEEE 802.15.4 has basically applied CSMA/CA mode competitive communication which is similar with IEEE802.11 [64]. It can be divided into beacon network (Beacon-enabled network) and non-beacon network (Non beacon-enabled network) [22]. Non-beacon network mainly uses un-slotted CSMA/CA instead of super-frame structure, because only super-frame structure has the concept of so-called back-off slot [22]. Whereas, beacon network uses super-frame structure and has the coordinator. The approach is known as slotted CSMA/CA. Beacon enabled network is usually used for energy efficiency [9]. As IEEE 802.15.4 provides beacon network and non-beacon network, non-beacon network is seen as an Ad hoc network, and all of the data transmission needs CSMA/CA to obtain the communication channel [22, 23].

IEEE 802.15.4 is mainly used in a device which power must be long lasting and has a low network throughput requirement, so that can make it in a very simple and low cost environment to use the wireless network to communicate with other devices [9, 64]. This feature also makes wireless sensor network become one of the main application areas of IEEE 802.15.4 LR-WPANs. It has several major advantages, such as easy to install, reliable data transfer, short distance of data transfer, very low cost and reasonable lasting power [8].

Following are the characteristics of the LR-WPANs [8, 9, 22, 24]:

- The transmission rates are 20kbps, 40kbps and it can reach up to 250kbps. If the radio frequency band is 868 – 868.8 MHz, the data transmission rate is 20 kbps, frequency band is 902 - 928 MHz, the data rate is 40 kbps, and frequency band is 2400 – 2483.5 MHz, the data rate can be 250 kbps.
- In the 2400 – 2483.5 MHz frequency band, it provides 16 channels. In the 902 – 928 MHz band, it provides 10 channels. In the 868 – 868.8 MHz band to provide a channel.

- It provides two kinds of topology: Star topology and Peer-to-Peer topology. It has two types of address modes: 16 bits or 64 bits, it can be used for power saving.
- It has guaranteed time slots (GTSs) ability, GTSs are mainly used to ensure the frequency bandwidth of the transmission, and it gives this device the ability for real time communication.
- It uses Carrier sense multiple access with collision avoidance (CSMA / CA) method to avoid data transmission collision.
- Provides acknowledged mechanism, it makes a more reliable data transmission.

ZigBee

IEEE 802.15.4 defines PHY layer and MAC layer [65]. PHY layer specifications defines it in the 2.4GHz frequency band with 250Kb/s transfer rate benchmark that works on low-power spread-spectrum radio substantive rules [8, 10, 23]. MAC layer specification defines multiple 802.15.4 radio signals and identifies how to share the air channel in the same work area [8, 9, 10]. However, it only defines the insufficiency of the PHY layer and MAC layer to ensure that different devices can communicate, also there on ZigBee alliance (www.zigbee.org) [12]. As mentioned before, IEEE 802.15.4 is the foundation of ZigBee network stack architecture, so ZigBee standard directly quotes the physical (PHY) layer and media access control (MAC) layer from IEEE 802.15.4 standard. However, ZigBee standard defines the network layer architecture above [66]. The first version of ZigBee standard was released in 2004 by the ZigBee Alliance and the objectives of ZigBee standard are short distance, low transmission rate, low cost, low power, low complexity, high reliability, and high scalability of wireless communication technology [8, 9, 10, 11, 12].

ZigBee protocol stack is simple, relatively easy to implement and less system

resources is required.The protocol stack reference model is shown in figure 2-5.

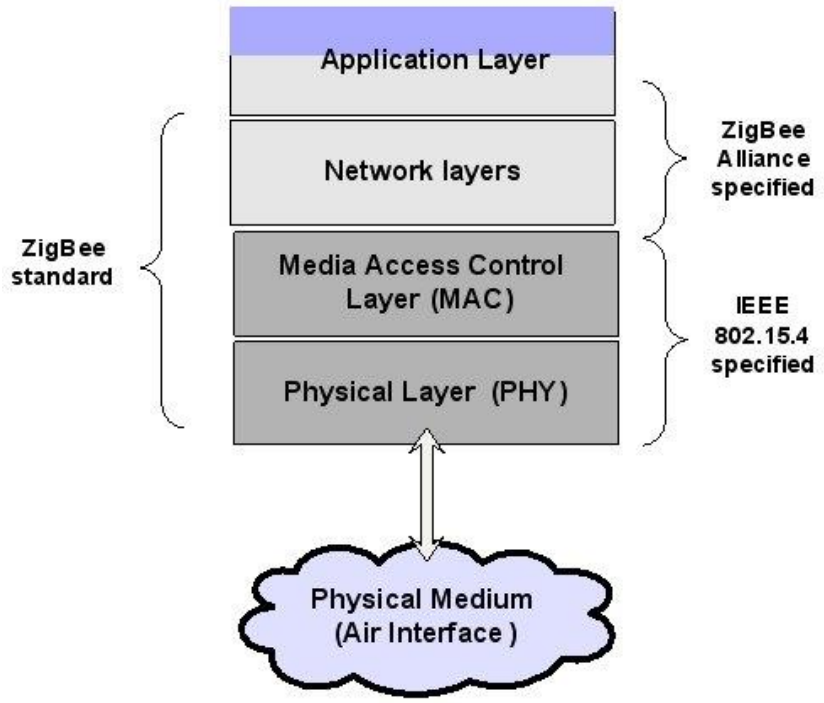

Figure 2-5: ZigBee Protocol Stack Model [23]

ZigBee network devices are divided into reduced-function device (RFD) and full-function device (FFD) [8, 9, 12]. FFD devices can be used as a router, coordinator, and an end device. It also can be responsible for controlling of the child nodes of communication, data collection and distribution control, etc [8, 66]. RFD devices can only be the network end devices, but cannot communicate directly with each other [8, 65]. RFD devices can only communicate through the FFD devices for sending and receiving data, and it does not have routing and relay functions [65]. ZigBee network has a very strong robustness and system reliability [65, 66].

The ZigBee network supports three types of wireless network topologies: Star, Mesh, and Cluster Tree, as illustrated in Figure 2-6 [12]. Depending on the application areas, actual environmental condition, network complexity, and reliability, different ZigBee network topologies will be used [23]. Firstly, star

topology is a low complexity network topology where all devices send message to coordinator for redirection [12]. In addition, it has the low signal latency [65]. The drawback of this topology is the limitation in network coverage area [66]. Secondly, cluster tree topology can be seen as a hierarchical version of star topology [12]. This is because it still uses direct point to point communication between two devices like star topology. Cluster tree topology is an intermediate complexity network topology [12]. Lastly, mesh network topology has multiple paths for each node communication in case of one single device failure [66]. For instance, if one router fails in the network, the disconnected device will be rejoined to the network by a new nearby router [66]. Therefore, mesh topology provides robustness and reliability to the network [23]. Mesh topology is also the most complex network topology compared with the other two [9].

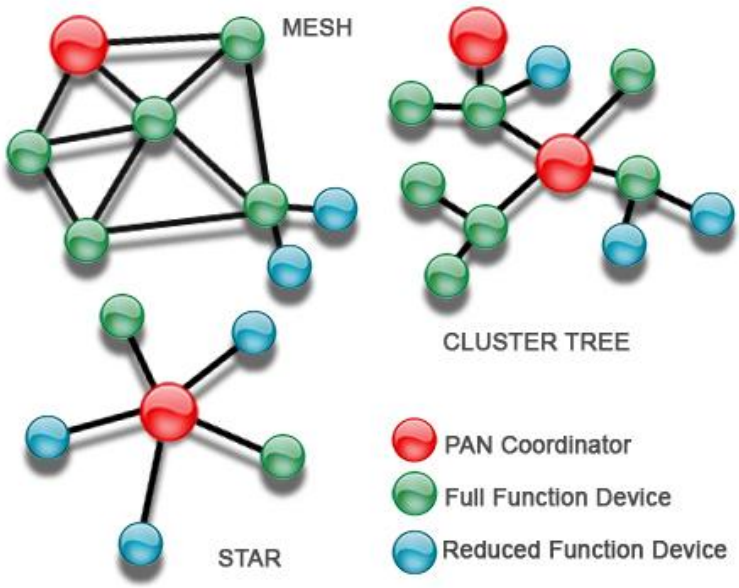

Figure 2-6: ZigBee Network Topologies [65]

The following are the main features of ZigBee standard [12, 23, 65, 66]:

- Use different radio frequency bands: are 2.4 GHz, 868 MHz, and 915 MHz, respectively.
- Low data transmission rate, the rate is between 10kbps-250kbps depending on the frequency bands. Maximum transmission rate for 2.4 GHz is 250kbps, 40kbps for 915 MHz, and 20kbps for 868 MHz.

- Small transfer range – it is normally around 10-75m.
- Low power consumption – ZigBee has a low transmission rate and low transmission power. As it uses sleep mode, a very low power is consumed. ZigBee devices can make power last from six months to two years by using only a pair of AA batteries.
- Low cost – because the architecture is simple, the cost is reduced.
- Large network capability – A star topology of a ZigBee network can accommodate up to 254 devices and a master device. Up to 100 ZigBee networks can exist within one area, and has flexible composition.
- Safety: ZigBee provides packet CRC integrity check function, supports authentication and certification, and uses 128 AES encryption algorithm.
- Use direct sequence spread spectrum (DSSS) modulation technique.

2.1.4.2 Operating System

With regards to the application area of the WSN, the desired WSN operating system is needed to efficiently use the limited memory of sensor nodes, low-speed low-power processors, sensors, low-speed communications equipment, limited power, and be able to provide maximum support for a variety of specific applications [20]. Under the support of sensor network operating system, multiple applications can concurrently use the system resources, such as computing, storage and communication [20].

In the sensor networks, a single sensor node has the following two outstanding features. One is intensive concurrency [25]. There may be several needs for simultaneously logic control. It requires the operating system to effectively fulfill this kind of logic control which happens frequently, with a high degree of concurrency [25]. The other one needs the operating system to allow the applications to easily control the hardware, and ensures that the various parts of

the application can be convenient to re-group [25]. The features above have proposed new challenges to the design of the operating system for sensor networks. TinyOS [21] operating system was developed by the University of California, Berkeley in cooperation with Intel Research. It is basically able to meet these particular requirements. There are also some other operating systems, such as MANTIS OS [28] developed by Colorado University, SOS [29, 49] operating system developed by UCLA, T-kernel [30], and Contiki [31]. However, the TinyOS operating system is the most popular operating system as it has been used by more than 400 research institutions around the world [21].

TinyOS is an open source and component based operating system and it is a new operating system for sensor networks, especially for resource constrained wireless devices [20]. It was originally written in C programming language, but the researchers found that C language was not effective or convenient for sensor network applications and development of operating system [21]. Therefore, after carefully researched and designed, the C language was extended and the nesC programming language was proposed [21].

2.1.4.3 Programming Language

TinyOS operating system, libraries and applications are all written in nesC language [20]. nesC [21] is a new programming language for the component-based and event-driven applications. nesC language is mainly used in network embedded systems such as sensor networks and is similar to C programming language which is an extension of the C language [67]. nesC can link other software components together to form a robust network embedded system, but it cannot dynamically assign the memory [21].

2.1.5 Wireless Sensor Network MAC Protocol

2.1.5.1 The characteristics of MAC protocol for wireless sensor network

Wireless sensor network is different from traditional wireless voice and data network in terms of the limited power capacity, the node self-organization, huge number of nodes, and unexpected volume of businesses, etc [65]. As a result, these wireless sensor network features show that the design of the WSN MAC protocol should be different from the traditional MAC protocols.

2.1.5.2 The important factors for wireless sensor network MAC protocol design

For the designing of a good wireless sensor network MAC protocol, the following factors need to be considered:

- **Energy Efficiency**

  In wireless sensor network MAC protocol performance, energy efficiency is the most important factor [63]. As stated in the previous section, the energy of sensor nodes is very limited. In the actual design of the wireless sensor network, the node cost will be reduced and the node for one-time use is set as a design goal [3]. Moreover, wireless sensor network may work in rural areas or dangerous environments, which will become difficult to replace batteries or recharge the batteries for sensor nodes. Therefore, in the design stage of the wireless sensor network, extending the survival time of the sensor nodes needs to be considered [8].

- **Scalability**

  MAC protocol Scalability refers to a MAC protocol that adapts to the network size, network topology structure, and network node density changes [68]. Wireless sensor network is a dynamic network and sensor nodes have mobility [9]. There are many factors affecting the size of the network and topology structure. Therefore, a good MAC protocol should enable the

network to adapt to these changes quickly and efficiently.

- **Collision Avoidance**

  To avoid channel collision is a fundamental task of the MAC protocol, which determines the nodes when and how to access the shared transmission media and transmit data [68].

- **Channel Utilization**

  Channel utilization reflects how the channel bandwidth is used during network communication [63]. In the cellular mobile communication system and wireless LAN, channel utilization is a very important performance indicator [59]. This is because in this system, the bandwidth is a great significant resource and the system needs as much as possible to accommodate more traffic. In contrast, in wireless sensor network, the number of nodes in communication is determined by the application task, so channel utilization performance in wireless sensor network is in second place [63].

- **Network Delay**

  Network delay refers to the time interval during the sender that sends a packet until the receiver successfully receives the packet. In wireless sensor network, the importance of delay mainly depends on the application of the network [68].

- **Network Throughput**

  Network throughput represents the successful received data quantity which the sender sends in a given period of time. Many factors can affect the network throughput, for example, the effectiveness of collision avoidance mechanism, channel utilization, delay and control overhead, etc [68].

Therefore, in order to design a wireless sensor network MAC protocol, the main consideration should be given to, one is the energy efficiency, and the other is the scalability, and some other performance indicators including network throughput, delay, channel bandwidth utilization which are the primary consideration in traditional MAC protocol [63, 68].

2.2 Simulation Platforms for WSN

Conducting the WSN experiments on physical hardware may invite a number of issues such as financial investment, level of difficulty to implement, time consuming, test environment factor and potential negative impact of hardware failure [4].In contrast, WSN simulation tools are cheaper, easy to setup, reliable, flexible, realistic, efficient and accurate [13]. Therefore, WSN simulation technology has been a hot topic and attracted an intense research in the past few years [16]. WSN simulators have been playing a great role in promoting the development of wireless sensor network technologies [41]. For example, NS2 network simulator [19] uses a series of object-oriented design methods, through a large number of simulation modules which provide the extensive uses of simulation analysis on network protocols and achieve very intuitive system performance analysis. There are also some similar simulators, such as SENS [34] and SensorSim [36], etc. It has been confirmed that there are huge majority of research papers on WSN algorithms, protocols, security, and performance analysis that come from simulation experiment results. A few of them will be identified here [5, 22, 23, 37, 38, 39, 40].

The emergence of wireless sensor network has opened up many new application areas, but it also brings many new problems which have not yet seen in the wired and wireless networks in the past. In the wired sensor network simulation, the widely used simulation tools such as NS2 [19] and OPNET [53] are not very suitable for performance analysis in wireless sensor network. So now there are some new simulation languages and simulators developing for WSN [41].

For the most of computer networks, when we need to verify and analyze the network solutions, there are usually three categories - analytical method, experimental method, and simulation method [39].

- Analytical method – it is based on the limited conditions and reasonable

assumptions, including describing research objects and systems, and analysis of abstract mathematical model.

- Experimental method –design the required hardware, software and configuration set up a real test-bed for research on network protocol, behaviour, and performance.
- Simulation method –apply network simulation software to set up network and system models, and then run the models to do calculations and analysis.

In this thesis, both experimental and simulation methods will be used. The goal of this chapter is to describe the characteristics of WSN simulation, different kind of WSN simulators and their features, strength, and weakness.

2.2.1 The Characteristics of WSN Simulation

Wireless sensor network is a highly application-oriented network type. The characteristics of the simulation in the following areas significantly are different from the existing wired and wireless network simulation.

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 [39]. So, in the WSN simulation scale, a large number of nodes in parallel computing must be considered [1, 42].

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 [1]. In contrast, in the previous network model, the life and energy consumption analysis of the node are the most important objectives of the analysis [39, 43].

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 [39]. 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 [1, 42].

2.2.2 The Development of WSN Simulator

Nowadays, a vast number of WSN simulators have been developed. Based on the functions of WSN simulators, they are usually divided into three levels [16, 44].

a) Algorithm Level –simulators in the algorithm level mainly focus on simulating the wireless sensor network data structure, logic and presentation of the algorithms, for instance, NetTopo [16], TRMSim [69], and AlgoSensim [70].
b) Packet Level – simulators in the packet level simulate the physical, data-link, and network layers in the standard OSI network stack, such as NS-2 [19], SensorSim [36], SENS [34] and so on.
c) Instruction Level –simulators that fall into this level usually are the hardware emulation, like simulating CPU execution, radio chip, ADC, and LED [44]. Generally, they simulate particular hardware sensor platform and they are often regarded to as emulators [39]. They compute the power of a particular sensor’s hardware platform in WSNs, for example, Avrora [13], Atemu, [47]

and TOSSIM [20].

2.2.2.1 TOSSIM

TOSSIM (TinyOS Mote Simulator) [20, 21, 33] is a discrete event simulator used to simulate wireless sensor network. TOSSIM can simulate the entire TinyOS applications achieved by replacing implementation parts of the components [21]. TOSSIM can directly compile the TinyOS NesC code to the executable file which can run in the PC environment [20]. So TOSSIM provides a platform to test the program, so that downloading the program to the real Motes is not required.

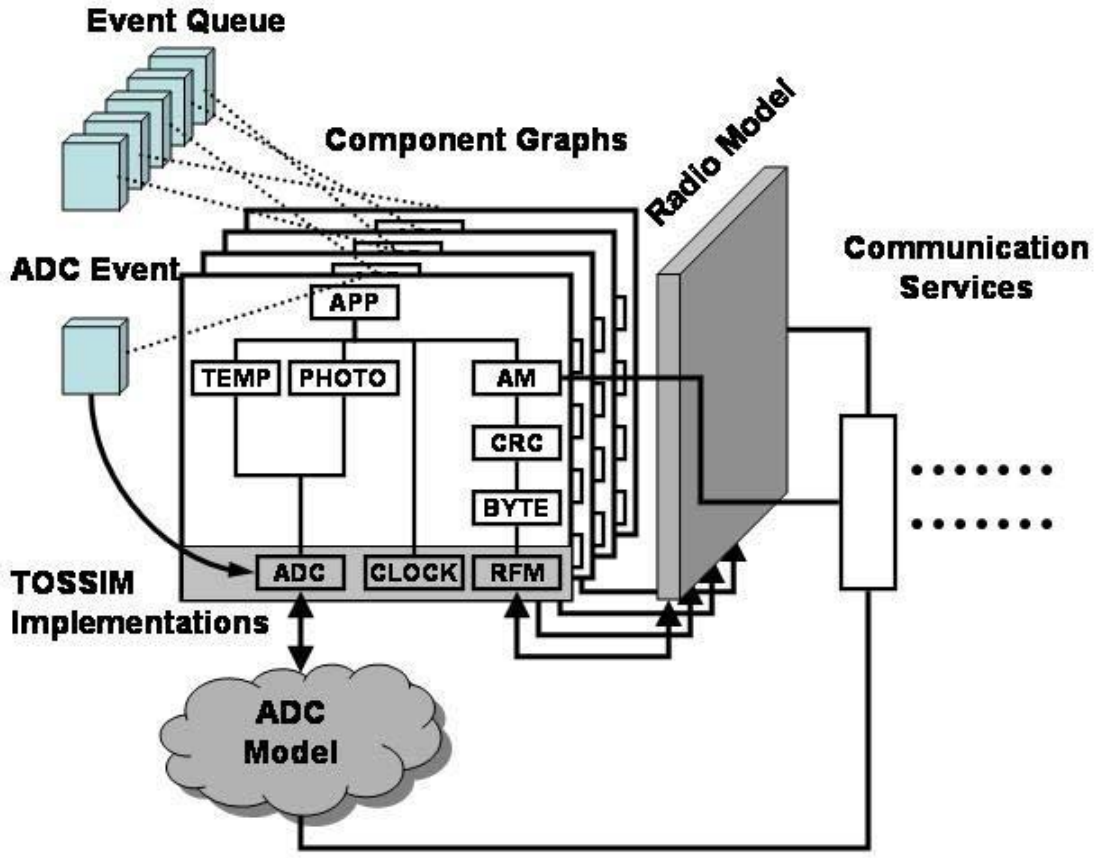

Figure 2-7- TOSSIM Architecture [20]

Figure 2-7 shows the working flow of TOSSIM. The TOSSIM architecture is consisted of 5 segments: Frames, Components, Models, Services and Events.

TOSSIM is a TinyOS library [21]. The core code is located in /tos/lib/tossim directory. Each TinyOS source code directory has an optional sim subdirectory which contains the package for the simulation [20]. For example, in the /tos/chips/atm128/timer/sim directory that contains Atmega 128 timer package for TOSSIM simulation. To compile TOSSIM simulation, the Make command is needed under the optional sim directory, typing: $ make micaz sim.

The following list is the key features and limitations of the TOSSIM simulator [21, 33]:

- TOSSIM is a library: a program needs to be written to configure the simulation. Run it.
- TOSSIM supports two types of programming interfaces: Python and C++. Python is powerful and allows you to run the simulation carried out with the dynamic interaction, debugging. However, Python interpreter is a performance bottleneck. Typically, C++ programming interface can simply transform one type of code to another type of code.
- TOSSIM can simultaneously simulate thousands of nodes. Each node in the simulation runs the same TinyOS program.
- Currently, Micaz mote is the only supported platform of TOSSIM.
- TOSSIM does not provide energy model, so it cannot evaluate the effectiveness of energy consumptions.
- TOSSIM only focuses on studying the behaviour of the nodes within specific TinyOS applications.
- It falls short when it comes to study the hardware operation of the network elements.

2.2.2.2 Avrora

Avrora [13, 14, 39] is a simulation tool based on the model and specialized for AVR single chip microcontroller program on Amtel and Mica2 platform. Avrora is built in JAVA programming language and was developed by the UCLA (University of California Los Angeles) Compilers Group [13].The following are the main features of Avrora [13, 14, 39]:

- Avrora was designed to simulate the behaviour of the specific hardware.
- Avrora provides cycle accurate level simulation for the AVR microcontroller, so it makes static program run accurately.
- It can simulate chip-on device drivers, and provides the regulated interface for chip-off program.
- Avrora provides flexibility and scalability of simulation, you can add monitoring code to report the performance of simulation, or generate reports after collecting statistics at the end of simulation.
- It provides a basic set of monitors which help analyze program execution and resource consumption, etc.
- Avrora provides a tool for analyzing power consumption, and can set the power consumption level for the particular device.

On the other hand, the downside of Avrora is because it is not good to be used to simulate the data-link, network, and application layer protocols [39]. In addition, it cannot provide a graphical user interface (GUI).

2.2.2.3 NS-2

NS-2 (Network Simulator-2) [19, 32, 33] is a well-known network simulator for discrete event simulation. 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 [19]. 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 [33]. 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 [16]. 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 [19].

2.2.2.4 OPNET

OPNET [19, 35] is a commercial communication network simulation platform. OPNET uses the network node and the process of the three-tier model to achieve the simulation of network behaviour. The wireless model is based on pipeline architecture to determine the connection and communication between nodes. OPNET users can specify the frequency, bandwidth, power, and include the antenna gain patterns and terrain models and some other features. OPNET provides a number of models, including TCP/IP, 802.11, 3G, etc. In addition, some researchers have already implemented the TinyOS's NesC simulation on OPNET. However, in order to achieve the simulation of wireless sensor networks on OPNET, it needs to add energy model, and plus OPNET itself, which will be more focused on performance evaluation of QoS. Plus, it also suffers the same problem of scalability like NS-2 [19].

2.2.2.5 SENS

SENS [34] was developed to solve the deficiencies of the traditional network

simulators, whichhas been used in the field of wireless sensor network. It came after NS-2 providing some necessary changes [39]. 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.

2.2.2.6 SensorSim

SensorSim [36] was developed based on NS-2 simulator to create wireless sensor network model library. SensorSim uses sensor function models, power models, and radio propagation models, respectively to simulate the node software (e.g. various protocol stack, application-program) and the node's hardware parts (e.g. CPU, power supply, RF receiver circuit and sensors) [16]. The main idea of hardware simulation is to build a single multi-mapping relationship between power supply and a variety of physical hardware, through analyze the characteristics of the hardware power usage to summarize the node power consumption problems.

2.2.2.7 NetTopo

NetTopo was designed for testing and investigating algorithms and protocols of wireless sensor network, and provided both simulation and visualization functions [16, 17, 18]. NetTopo is open-source software which can be downloaded from SourceForge [16] or its official website [19]. NetTopo was developed in JAVA programming language. It can be used to simulate very large scale networks.

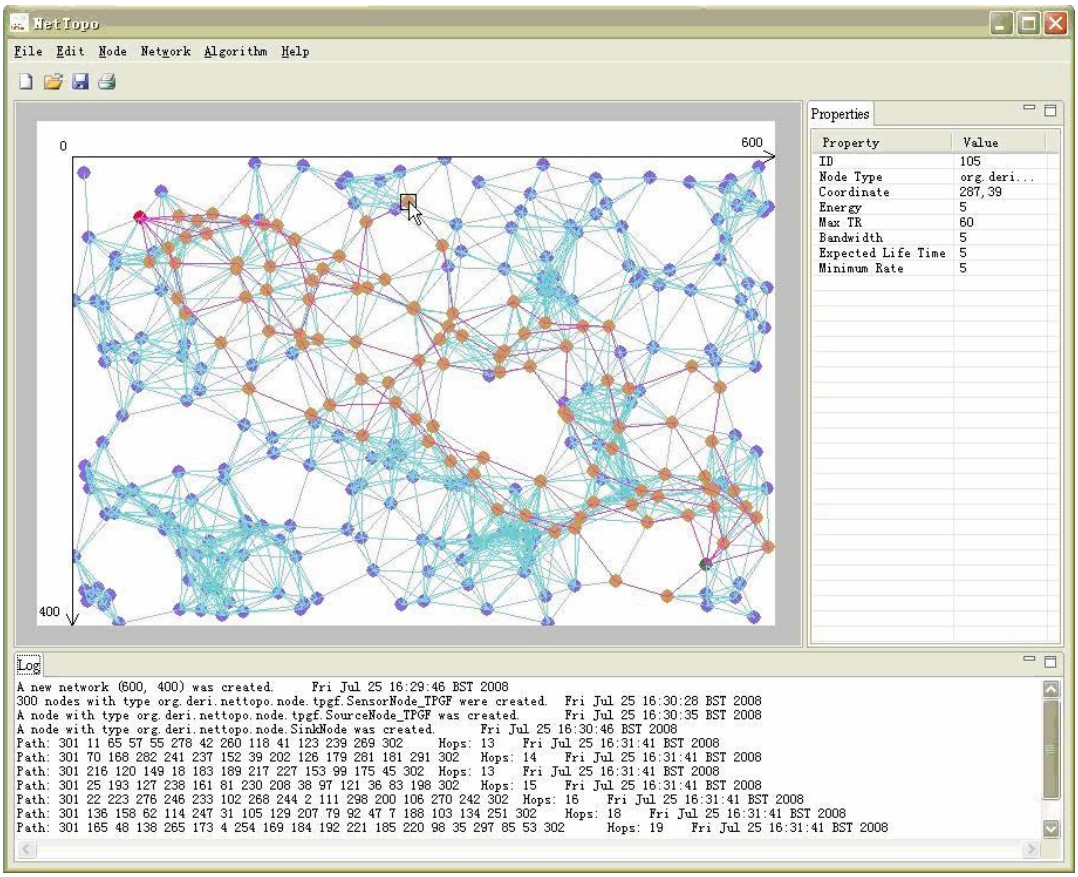

Figure 2-8- NetTopo Graphical User Interface [16]

NetTopo is JAVA –based simulator and needs Eclipse to run the simulation [17]. In this thesis, Eclipse software was installed in the virtual machine and run and tested NetTopo. Unfortunately, this simulator still has some bugs and was not realistic to use for simulating of a small scale networks. It was very similar to another WSN simulator such as TRMSim [69] which has also been tested in this study.

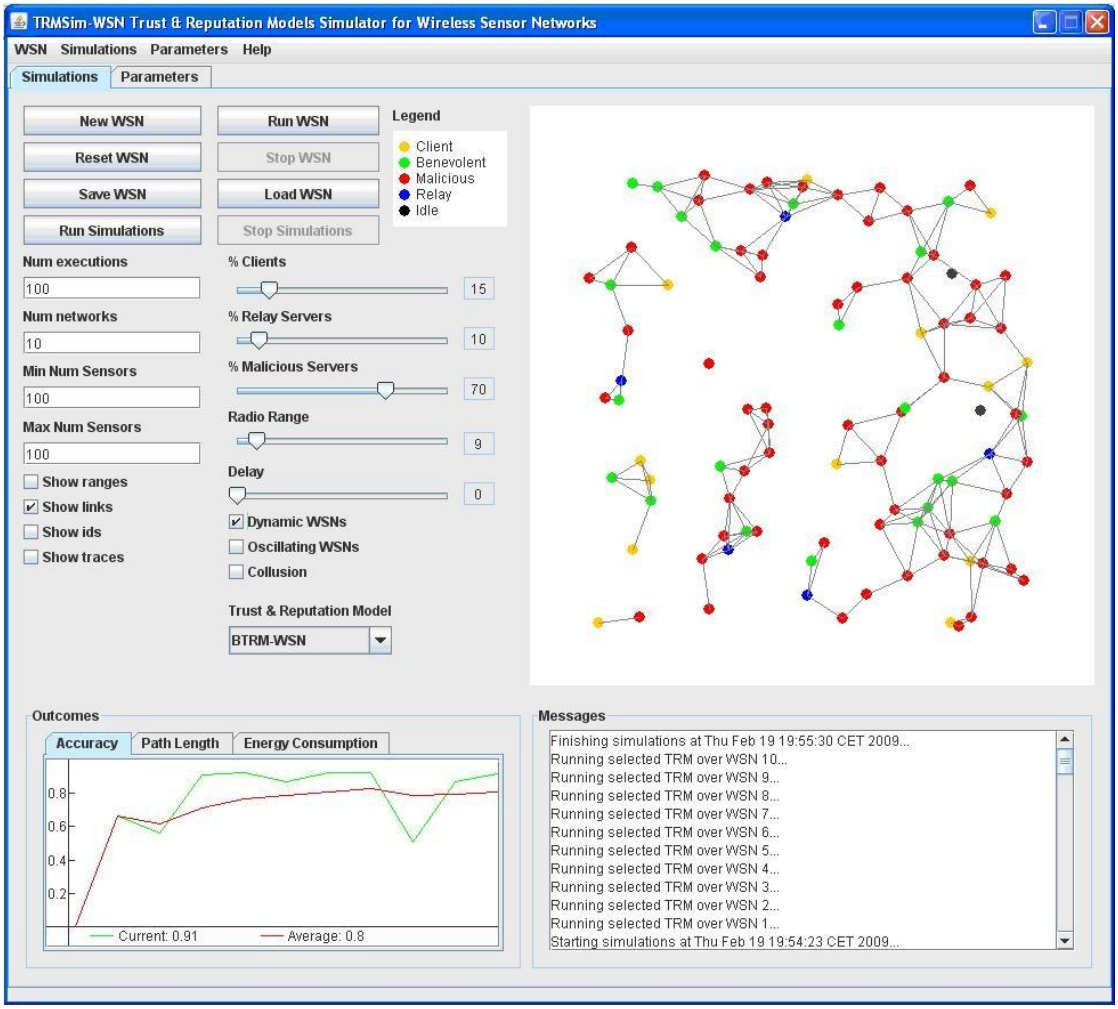

Figure 2-9: TRMSim Graphic User Interface [69]

2.2.3 OMNET++

2.2.3.1 OMNET++ Introduction

OMNET++ [3, 19] is a discrete event simulatorthatprovides for processing and event-driven simulation. OMNET++ is the abbreviation for Objective Modular Network Test bed in C++. It is an open source, component-based, modular, and communication network simulation platform [4]. The development of OMNET++ is as popular as the open source Linux operating system. In addition, it has been used broadly in the field of science and industry in recent years, such as in many Universities and companies [3]. Now, OMNET++ is divided into commercial

(OMNEST) and non-commercial two versions. Non-commercial version is mainly used for academic research with open source, thus, everyone can make the improvement necessary to this software. Currently, the latest version is OMNET4.1.

OMNET++ has a powerful graphical interface and embedded simulation kernel [16]. Wireless sensor network design requires considering the impact of energy efficiency, fault tolerance, synchronization, quality of service, scheduling methods and system topology [5]. Comparing with the NS2 and OPNET simulation platform, OMNET++ can run on multiple operating systems (Windows, Linux, Mac OS/X), easily defines the network topology, quality of service and possesses programming, debugging and tracing functions [3]. The main application field is the simulation of communication network and distributed systems [71]. Moreover, it can be widely used in the simulation of complex areas of IT systems as the advantage of its flexible structure [71].

- **OMNET++ application fields**

It can be used in various application fields [3, 5, 19, 71]:

- Simulate wired and wireless communication network traffic modeling
- Simulate Protocol modeling
- Simulate Queuing networks
- Simulate multithreaded processors and other distributed hardware systems
- Verify the effectiveness of hardware architecture
- Evaluate the performance of complex software systems
- Simulate other applicable fixed event systems

OMNET++ simulation is divided into two levels [5]. One is based on the NED programming language level, defining the simulation modules and network [5]. The other level is based on C# and .Net programming language level to define

module, network's behaviour and actions [5]. OMNET++ provides users with effective tools to describe real system structure and main features including hierarchical embedded modules, module type examples, flexible module parameters and channel-based mechanism for message delivery, etc [4].

The following are the main development platform of OMNET++ [71]:

- Solaris, Linux system (or other kind of Unix systems)
- Cygwin32 and Win32 (with Win32 interface gcc compiler tools)
- Microsoft Visual C++

2.2.3.2 OMNET++ Framework

OMNET++ mainly consists of six parts which are simulation kernel library, network topology description language compiler (NED), graphical network editor (OMNET++IDE), simulation program graphical user interface (Tkenv), simulation program command-line user interface (Cmdenv), utilities (graphical output tools - plove and scalar, make file creation tool, etc) [71]. Simulation kernel library occupies the most important core position in OMENT++ [4]. All of user-written simulation programs need to connect with simulation kernel library [4]. The following are the details of the other two important components of OMNET++:

a) The NED language

NED is a modular network description language [21]. Network description includes a description of a large number of the components, such as channel, simple and compound modules.Description of these components can be used to describe the various networks [21]. NED language is used to define the network topology of the model [5]. Relatively simple network topology can use graphical NED to define, but the complicated network topology description should be written by NED source file [21].

b) User Interfaces

OMNET++ user interface is used to implement the interaction between human and computer [19]. It allows users to start and terminate the simulation, and also to change the internal variables of model [71]. OMNET++ graphical user interface is an excellent tool for users to understand well the internal operation mechanism of model [19]. The interaction between simulation kernel and user interface undertakes via a defined interface. Without changing the simulation kernel library and the model file, the simulation model can run on different interfaces [71].

OMNET++ currently supports two sorts of user interfaces, Tkenv and Cmdenv [19]. The simulation testing and debugging can be carried out in Tkenv interface. Tkenv is an easy to use window-based graphical user interface and supports tracking, debugging and implementing simulation functions. The main features of Tkenv are as follows [4, 19, 71]:

- ✓ Text output of each module has its own independent window.
- ✓ Auto biography message can be seen in Tkenv window during simulation process.
- ✓ Support simulation animation, mark the breakpoint.
- ✓ With a check window, you can check and change the variables of the model.
- ✓ Support graphical display of simulation results and the simulation results can use the bar chart and time sequence diagram shows.
- ✓ Simulation can repeat many times and snapshot files are used to display the model details.

Cmdenv interface is used for actual simulation experiment, because it supports batch processing. Cmdenv is a simple small command-line interface with fast implementation speed. It can run on all operating system platforms.

2.2.3.3 OMNET++ Language

OMNET++ is a discrete event simulator which provides for process and event-driven simulation [3]. OMNET++ is an abbreviation for Objective Modular Network Test bed in C++ [71]. It is an open source, component-based and modular open network simulation platform.

2.2.3.4 OMNET++ Installation

OMNET++ Supported Platforms

1. Windows 32-bit platform (Windows XP, Vista, 7)
2. Linux x86 32-bit/64-bit (Ubuntu, Fedora)
3. Some other Unix-like systems (Solaris, FreeBSD)
4. Mac 32-bit (Mac OSX 10.5,10.6)

This theis only concentrates on the simulation in the Linux platform (Ubuntu server 10.04 LTS). The following is a brief version of OMNET++ installation steps for research:

a) Because OMNET++ needs several packages to support running, including C++ compiler (gcc), java runtime environment (JDK 1.6), and some other related programs [5]. It is easy to install gcc and JDK in Ubuntu linux. Open a terminal and use administrator permission to get and install these packages with the following command: sudo apt-get install gcc, sudo apt-get install sun-java6-jdk.

b) Download the archive file, omnetpp-4.1-src.tgz from the official OMNET++ website http://www.omnetpp.org. Extract the archive to the location where you want to install and it may be your home folder /home/*Alex (your name)*, or directly extract to root folder /root.Use the following command to extract the archive file:tar xvfz omnetpp-4.1-src.tgz /home/Alex, then a sub directory will be created under /home/Alex, named *omnetpp-4.1*.Note that the path cannot

contain space.

c) Set environment variables by using following command to edit *.bashrc* file, sudo vi ~/.bashrc, and add the following three command lines to the file and save it:

$ exportPATH=$PATH:$HOME/omnetpp-4.1/bin,

$ export JAVA_HOME=/usr/local/jdk1.6.0,

$ export PATH=$JAVA_HOME/bin: $PATH.

d) After extracting the folder in the root directory, run the following commands in the terminal to configure the setting and verify the installation:

- ✓ $ ./configure (this command is to check the environment setting of library files.)
- ✓ $ cd /omnetpp-4.1/samples/dyna (enter dyna sample simulation folder)

  $./dyna (run sample simulation)
- ✓ make (Re-compile the system)

2.2.4 Castalia

This section provides an overview of Castalia simulator and briefly highlights Castalia, including its system requirements and installation. The main components of Castalia, such as structure, modeling, language, the dependency between Castalia and OMNET++, will be identified.

2.2.4.1 Castalia Introduction

Castalia [41, 48] is a robust simulator designed for Wireless Sensor Networks (WSN), Body Area Networks (BAN) and low-power embedded devices networks. Castalia is built on the famous event-driven simulator OMNET++ [41]. It has been

broadly used to test and validate the algorithms and protocols by reason of the reliable and realistic wireless sensor node behaviour, such as radio module and wireless channel, etc [48]. Researchers can easily import their algorithms or protocols into Castalia, and then merge with other provided features [41]. Castalia also support researchers and developers to build their own protocols. However, Castalia is not the tool for testing the code and hardware simulation for a particular sensor node like Avrora and Atemu. The following is the main features of Castalia simulator [41]:

- *Advanced channel model based on empirically measured data.*
  - ✓ *Model defines a map of path loss, not simply connections between nodes*
  - ✓ *Complex model for temporal variation of path loss*
  - ✓ *Fully supports mobility of the nodes*
  - ✓ *Interference is handled as received signal strength, not as separate feature*
- *Advanced radio modelbased on real radios for low-power communication.*
  - ✓ *Probability of reception based on SINR, packet size, modulation type. PSK FSK supported, custom modulation allowed by defining SNR-BER curve.*
  - ✓ *Multiple TX power levels with individual node variations allowed*
  - ✓ *States with different power consumption and delays switching between them*
  - ✓ *Realistic modeling of RSSI and carrier sensing*
- *Extended sensingmodeling provisions*
  - ✓ *Highly flexible physical process model.*
  - ✓ *Sensing device noise, bias, and power consumption.*
- *Node clock drift.*

- *MAC and routing protocols available.*
- *Designed for adaptationand expansion.* [41]

As Castalia is based on OMNET++, some of its features come from OMNET++ platform, including reliability, running speed, and modularity.

2.2.4.2 Castalia Structure

For getting realistic node behaviour, Castalia simulates all the components of real sensor nodes. Figure 2-4 shows the internal structure of Castalia node module. From the figure, we can see that the node module is composed by several device modules including application, sensor manager, resource manager, mobility manager, and communications composite modules. In this figure, the solid arrows represent the message passing between modules and the dashed arrows represent the function call. As we see, many modules call the resource manager for simulating the energy consumption. The application, routing, and MAC modules are the most frequently changed by the user for implementing new algorithms and protocols [41].

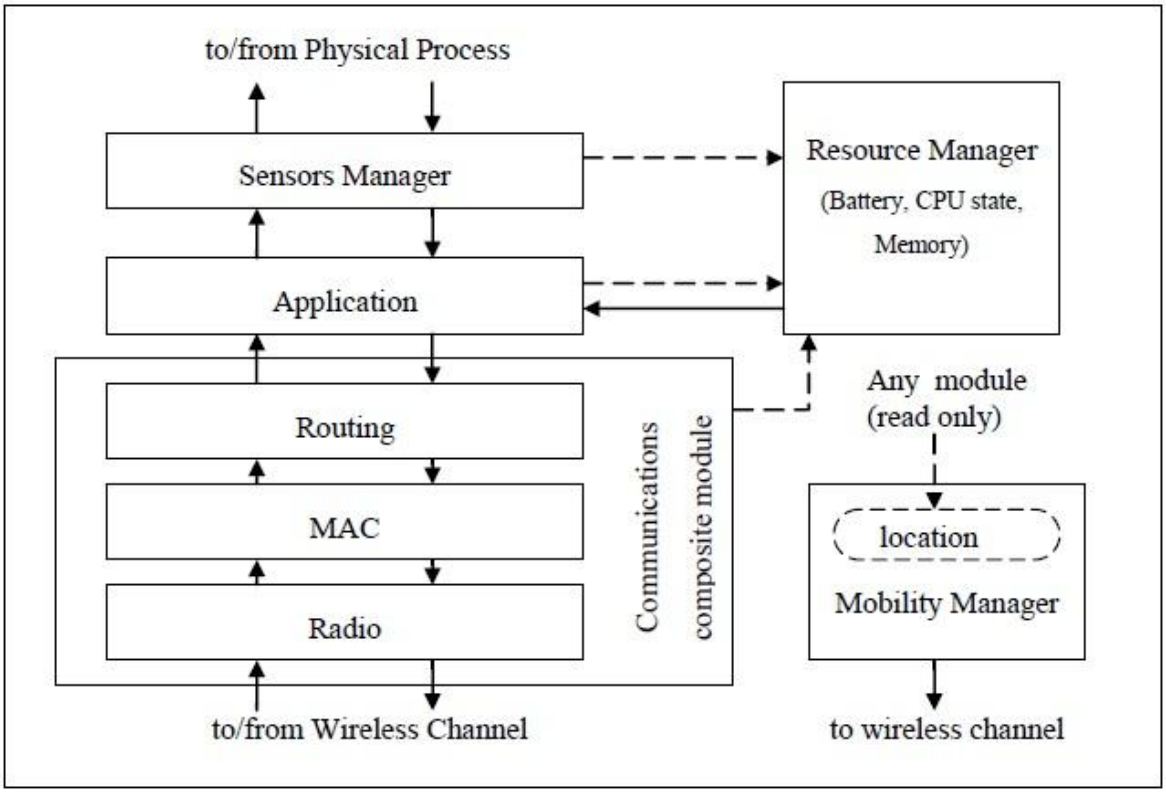

Figure 2-10: Castalia Node Module Structure [41]

2.2.4.3 Castalia Language

Castalia is well-known based on OMNET++, therefore NED language is used as well. By using NED language, users can easily modify modules to create their desired one, for example, to modify the module parameters, name, interface, and even sub-module [5]. In the Castalia directory, files end with ".ned" has NED language code inside used to define the module. There are also some C++ language codes used in Castalia to define node behaviour [41, 48].

2.2.4.4 Castalia Modeling

This section briefly describes the Castalia models. The main emphasis of Castalia simulator focuses on the communications composite module from the wireless channel module to the realistic radio module behaviour and the comparison of different MAC protocols [48]. The key points of each modeling in Castalia will be

discussed.

The Wireless Channel

The Castalia wireless channel model has been created by based on hundreds of thousands experimental test-beds measurements, and also includes average path loss modeling and the temporal variation behaviour [48]. According to the path loss, temporal variation, and the transmission power of the sensor node, the wireless channel model can be used to calculate the signal strength received from the transmitting node. Therefore, [41] stated that "Castalia is the most realistic simulator one could find for WSN and BAN".

The following formula is used to calculate the average path loss between two nodes.

$$PL(d) = PL(d_0) + 10 \cdot \eta \cdot \log\left(\frac{d}{d_0}\right) + X_\sigma$$

PL(***d***) is the path loss at distance **d**, PL(***d*0**) is the known path loss at a reference distance ***d*0**. **η** is the path loss exponent, and **Xσ** is a gaussian zero-mean random variable with standard deviation **σ**. The four parameters in italics and bold, are defined as parameters of the wireless channel module [41].

Parameter	Default (Realistic)	Description
η	2.4	Pass loss exponent
PL(*d*0)	55	The path loss at a reference distance *d*0in dBm
***d*0**	1.0	Meters
σ	4.0	Sigma, in the IDEAL mode σ=0
Collision Model	2	Interference model=2 (Additive collision model) Interference model=0 (no collision)
Threshold	-100	Signal delivery threshold in dBm

Table 2-1: Wireless Channel Parameters

The wireless channel model also has some other functions such as allowing the

node mobility, sending signals to the radio model and providing simpler models and so on.

The Radio

Castalia radio model can simulate a lot of features of a real low power radio [48]. There are three types of radio provided in Castalia simulator, CC2420, CC1000, and BAN (Body Area Network) radio [41]. In this thesis, simulation only for CC2420 will be examined. The following outlines a number of main features of Castalia radio model [41, 48]:

- Provides multiple radio states: receive, transmit, listen, and sleep states. In addition, sleep state is configurable.
- Supports vary levels of transmission power.
- Provides vary power consumption for the different levels of transmission power and different states.
- Supports varymodulation modes (FSK, PSK, DiffQPSK, DiffBPSK, and IDEAL modulation). User can also define his/her own modulation.
- Supports the calculation of RSSI (Received Signal Strength Indicator)
- Supports the calculation of fine-grain interference and the bit errors calculation
- Supports CCA (Channel Clear Assessment) and MAC model interruptions.

RX Modes		
Parameters	Default (Realistic)	IDEAL
Data Rate	250kbps	250kbps
Modulation Type	PSK	IDEAL
Bits Per Symbol	4.0	4.0
Bandwidth	20MHz	20MHz
Noise Bandwidth	194MHz	194MHz

Noise Floor	-100dBm	-100dBm
Sensitivity	-95dBm	-95dBm
Power Consumed	62mW	62mW

Table 2-2: Radio Receive Modes

The following table shows the eight possible transmission power levels and their corresponding energy consumption levels in Castalia. As you can see, the first row of the table lists the output power of the different transmission levels in dBm and the second row shows how much energy the radio is spending when transmitting in a power level.

TX Levels	
TX_dBm	TX_mW
0	57.42
-1	55.18
-3	50.69
-5	46.20
-7	42.24
-10	36.30
-15	32.67
-25	29.04

Table 2-3: Radio Power Transmission Levels

MAC

The Medium Access Control (MAC) protocol plays a pivotal role of the wireless sensor node behaviour. Castalia provides four sorts of MAC models [41]:

- Tunable MAC
- T-MAC/S-MAC
- IEEE 802.15.4 MAC
- Baseline BAN MAC

Tunable MAC

Tunable MAC was the initial motivation of the author to develop Castalia simulator for testing tunable MAC protocol in a realistic channel radio model [41]. Tunable MAC is a duty-cycled MAC protocol which supports CSMA/CA mechanism for data packets transmission. The following are parts of the crucial parameters which can be tuned in tunable MAC [41]:

- Duty Cycle –it is a critical parameter, because it can affect energy consumption by reducing the node listen time.
- Listen Interval – this parameter is for setting of the time for node staying on listen state each cycle. It is useful to set listen interval to small, so the latency of data transmission can be reduced. For the radio with 250kbps data transmission speed, listen interval time is usually between 5ms and 10ms. The default setting for this parameter is 10ms.
- Beacon Interval Fraction –this parameter is used to set the time to wake up the sleeping nodes.
- Re Tx Interval – Retransmission interval is the parameter to set interval between retransmissions.
- Tx All Packets In Free Channel – this parameter is a conditional parameter to check if the channel is free. If the free channel is found, then all packets will be transmitted. Otherwise, only one packet will be transmitted.

There are also some other important parameters that can be tuned, such as “sleepduringbackoff”, “CSMApersistance”, “backoffType”, “backoffBaseValue” and so on. The tunable MAC parameters can also be adjusted dynamically through application model.

T-MAC/S-MAC

T-MAC and S-MAC are two popular MAC protocols used for WSN [50]. S-MAC has the following major aspects of energy consumption: packet collision, control

packets, packet overhead, and idle listening. S-MAC uses the fixed duty cycle sleeping mode to reduce the sensor nodes energy waste. T-MAC is based on S-MAC, but T-MAC upgrades the energy waste function by using adaptive duty cycle according to the traffic needs [50]. In Castalia, T-MAC and S-MAC are combined into one which can be tuned to have both protocols' functionalities [41]. The following Figure 2-5 shows the scheme of the T-MAC and S-MAC protocols.

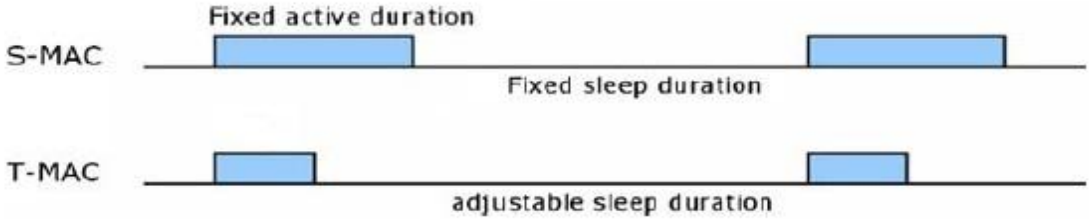

Figure 2-11: The scheme of T-MAC and S-MAC protocols [50]

There are also some important parameters can be tuned in T-MAC and S-MAC protocols, for example, maximum transmission retries, duration of each frame transmission, the time of listen state time out, the time of reply time out, packets resynchronization time, and collision resolution method [41].

IEEE 802.15.4 MAC

IEEE 802.15.4 MAC is defined by IEEE 802.15.4 wireless communication protocol standardthat has specified both PHY and MAC layers which mentioned in the earlier chapter. Castalia implements the main features of IEEE 802.15.4 MAC which includes CSMA/CA mechanism, Guaranteed Time Slots (GTS), Beacon-enabled PANs, Direct data transfer mode [48]. The following are some essential parameters in IEEE 802.15.4 MAC [41]:

- Enable Slotted CSMA – this parameter enables the slotted CSMA to regulate the channel access. The default value is always true in Castalia.
- Is FFD – it is used to set if the node is the full function device.
- Is PAN Coordinator – this parameter is to check if the node is a PAN

coordinator.

- Beacon Order – it specifies the duration of active and inactive full frame between beacons.
- MAC Max Frame Retries – this parameter defines the maximum number of retries before the packet is regarded as totally lost.
- Max Lost Beacons – it defines the max number of lost beacons.

Baseline BAN MAC

Castalia is also a simulator suitable for Body Area Network (BAN) and MAC protocol is an important part of the BAN simulation. Castalia implements baseline BAN MAC based on IEEE 802.15.6 standard draft proposal [41]. However, only the WSN related to MAC protocols need to be focused in this thesis, thus the BAN MAC will not be covered.

The Physical Process

Sensing is usually neglected in WSN simulators [16]. Usually, the simulation for sensing is to feed random numbers to the nodes, or each node to set a static value [16]. Castalia can capture some essential elements of sensing. The basis of the model is sources of values that their influence is diffused over space. The sources can change in time and space, in another word, change the position and value over time. The effect of multiple sources in a certain point is additive. More specifically the model that determines the value of the physical process at a certain location and at a certain time [41] is:

$$V(p,t) = \sum_{all\,sources\ i} \frac{V_i(t)}{(K \cdot d_i(p,t)+1)^a} + N(0,\sigma)$$

Where: *V(p,t)* denotes the value of the physical process at point p, at time t. *Vi(t)* denotes the value of the ithsource at time t. *di(p,t)* denotes the distance of point p from the isource at time t. *K and a* are parameters that determine how is the value from a source diffused. *N(0,σ)* is a zero-mean gaussian random variable with

standard deviation σ.K, a and σ are regular Castalia parameters as shown in below:

doublemultiplicative_k = default (0.25);

doubleattenuation_exp_a = default (1.0);

double sigma = default (0.2);

Vi(t) and *di(p,t)* are not directly given but they are calculated by the way we describe the behaviour of the sources. Up to five sources can be defined [41].

Castalia provides a one to one correspondence between sensing device type and a physical process module [48]. The physical process model supports only one output per point per time [41]. The simpler representation and implementation physical process model still can cover many practical requirements.

Routing

Castalia pays less attention on routing protocol compared with wireless channel, radio, and MAC modules [41]. This is because the initial purpose of Castalia simulator does not focus on routing. There are few routing protocols providing in Castalia and two routing protocols in the early version of Castalia – simple tree routing and multi path rings routing. After the redesign of Castalia simulator, only the multi path rings routing and bypass routing (which means no routing protocol is implemented) are supported since Castalia version 3.0 [41]. However, there is one paper [51] stated that the researcher implements CTP routing on Castalia simulator to evaluate its performance.

2.2.4.5 Castalia Installation

System Requirements

As mentioned before, OMNET++ can be installed on Windows and Linux

operating systems or on Cygwin under Windows environment. On the other hand, Castalia is only supported in Linux or Cygwin platform instead of Windows platform. This is due to different Castalia version being matched with specific OMNET++ version. In this thesis, Castalia version 3.1 which is supported by OMNET++ version 4.0 or 4.1 will be utilized.

As already stated earlier, this thesis will focus on the simulation in the Linux platform (Ubuntu server 10.04 LTS) only. The following is the steps of Castalia installation:

1) Go to Castalia official website [41] and download the latest version of Castalia– Castalia-3.1.tar.gz.

2) Extract the archive to the location where you want to install, which may be your home folder /home/*Alex (your name)*, or directly extract to root folder /root. Use the following command to extract the archive file: tar xvfzCastalia-3.1.tar.gz /home/Alex. A sub directory will be created under /home/Alex, named *Castalia-3.1*.Note that the path cannot contain space.

3) Build your Castalia by using the following commands:
   - cd Castalia-3.1
   - ./make make

   It costs a little time for script running to generate a Make file. After the Make file is generated, enter *make* command in the command line to build Castalia.

4) Set an environment variable by adding Castalia/bin to the PATH. Use bash to edit *.bashrc* file, sudo vi ~/.bashrc, and add the following command to the file and save it:

   $ exportPATH=$PATH: $HOME/Castalia/bin

2.3 Related Work

The most similar to this thesis study in goals is [4]. In that paper, it evaluated the accuracy of the widely used OMNET++ simulator in WSN. The authors presented the effort to evaluate the reliability of OMNET++ by considering a simple experimental setup made of up to six Tmote Sky sensor nodes and tested on it the performance of the flooding algorithm. The results of the testbed were compared with the results of the simulations of the same scenarios on OMNET++. The authors also observed significant differences between the experimental and the simulated results; the simulations always showed better performance. In particular the results showed that high density of nodes and number of data sources strongly influences the performance of the flooding protocol in practice and this was not captured by simulations. The reasons of such results differences were analyzed and key parameters are determined and quantified by the authors.

CHAPTER 3. WSN HARDWARE

3.1 Wireless Modules

3.1.1 IRIS

Figure 3-1: IRIS module top view

Figure 3-2: IRIS module bottom view [73]

Size	**58 x 32 x 7 mm**
Processor	XM2110 based on Atmel ATmega 128-1
RAM	8 Kb
Frequency Range	2.4 – 2.48 GHz
Radio Transceiver	Atmel AT86RF230 802.15.4 compliant radio
Data Transmission Rate	250kbps
RF Power	3dBm Maximum
Transmission Range	Indoor more than 50m, outdoorup to 500m
I/O	51-pin expansion connector
Supported Interfaces	Digital I/O, Analog Inputs, SPI, I2C, UART
Supply Power	2.7 – 3.3 V

Table 3-1: IRIS Features [73]

3.1.2 MICA2

Figure 3-3: MICA2 module top view [74]

Size	**58 x 32 x 7 mm**
Processor	MPR400 based on Atmel ATmega 128L
RAM	4 Kb
Frequency Range	868/916 MHz
Radio Transceiver	Chipcon868/916 MHz multi-channel
Data Transmission Rate	38.4kbps
RF Power	-20dBm to 5dBm
Transmission Range	Up to 150m outdoor
I/O	51-pin expansion connector
Supported Interfaces	Digital I/O, Analog Inputs, SPI, I2C, UART
Supply Power	2.7 – 3.3 V

Table 3-2: MICA2 Features [74]

3.1.3 MICAz

Figure 3-4: MICAz module top view [75]

Size	**58 x 32 x 7 mm**
Processor	MPR2400 based on Atmel ATmega 128L
RAM	4 Kb
Frequency Range	2.4-2.48GHz
Radio Transceiver	CC2420 802.15.4 compliant radio
Data Transmission Rate	250kbps
RF Power	-24dBm to 0dBm
Transmission Range	Indoor 20-30m, outdoor75-100m
I/O	51-pin expansion connector
Supported Interfaces	Digital I/O, Analog Inputs, SPI, I2C, UART
Supply Power	2.7 – 3.3 V

Table 3-3: MICAz Features [75]

3.1.4 TelosB

Figure 3-5: TelosB module top view [76]

Size	**65 x 31 x 6 mm**
Processor	MSP430
RAM	10 Kb
Frequency Range	2.4-2.48 GHz
Radio Transceiver	CC2420 802.15.4 radio transceiver
Data Transmission Rate	250kbps
RF Power	-24dBm to 0dBm
Transmission Range	Indoor 20-30m, outdoor75-100m
I/O	6-pin and 10-pin expansion connectors
Supported Interfaces	Digital I/O, Analog Inputs, SPI, I2C, UART
Supply Power	2.7 – 3.3 V

Table 3-4: TelosB Features [76]

3.1.5 Imote2

Figure 3-6: Imote2 module top view [77]

Size	**48 x 36 x 9 mm**
Processor	Intel PXA271
RAM	32Mb
Frequency Range	2.4-2.48 GHz
Radio Transceiver	CC2420 802.15.4 radiotransceiver
Data Transmission Rate	250kbps
RF Power	-24dBm to 0dBm
Transmission Range	outdoor75-125m
I/O	6-pin and 10-pin expansion connectors
Supported Interfaces	SDIO, GPIOs, SPI, I2C, UART
Supply Power	3.2–4.5V

Table 3-5: Imote2 Features [77]

3.2Test-Bed Hardware

In this experiment, the entire network nodes are built on IRIS platform– IRIS XM2110 WSN starter kit from CROSSBOW which includes two sensor nodes and one base station. Crossbow has dispatched over million sensors to different customers for commercial or research purpose in a variety of applications. In addition, Crossbow is a leading provider of WSN products in the world.

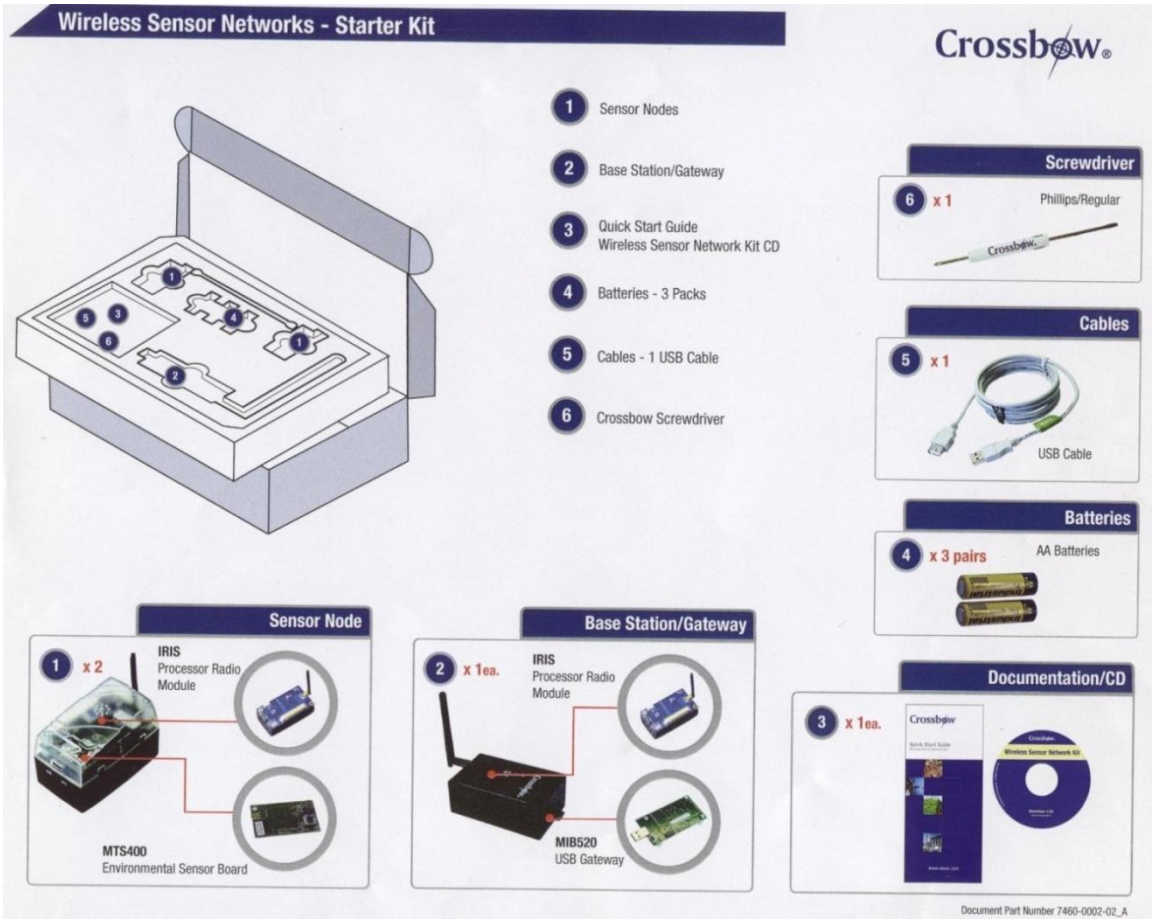

Figure 3-7 Crossbow WSN IRIS XM2110 Starter Kit

3.2.1 IRIS Process/Radio Board& MTS 400 Sensor Board

The sensor node includes two parts: IRIS process/radio board and MTS400 sensor board. The IRIS module is an ultra low powerwireless module and

contains IEEE 802.15.4/ZigBee compliant RF receiver- Atmel AT86RF230 [73]. IRIS module uses 16 channels and 2.4 GHZ frequency to provide 256kbps bandwidth two-way communication, through a pair of batteries and DC converter to provide a stable power, but other renewable power can be utilized to replace the currently used battery [73]. MTS400 sensor board has integrated radio, antenna, processor capability and a broad range of sensing capability temperature, pressure, light and humidity. The low-power consumption of IRIS module benefits from the low-power ATmega 1281 processor.

- ***Sensors***

There are a wide variety of sensors integrated in IRIS platform by using MTS400 sensor board, such as temperature, pressure, light, humidity, acceleration, magnetic and so on [73]. MTS400 sensor board was developed in conjunction with UC Berkeley and Intel Research Labs [73]. The operating temperature of MTS400 board is from -10°C to +60°C

- ***Processor***

The low-power consumption of IRIS module is due to the low-power ATmega 1281 processor, it provides 8kb RAM, 128kb program flash memory, 512kb measurement flash [73]. The ATmega 1281 processor only consume 8mA and 8μA in active mode and sleep mode, respectively.

- ***I/O Interfaces***

A wide range of interfaces are supported by 51-pin expansion connector like UART interface for serial communication and some other interfaces – SPI, Digital I/O, I2C, Analog inputs [73]. So these available interfaces make IRIS module easy to connect to different external peripherals.

- ***Power and Voltage***

IRIS module is powered by two AA batteries. The external power voltage is

between 2.7V – 3.3V, the process uses a 10 bit ADC (Analog to Digital Converter) module to convert the analog input to its 10-bit digital data [73].

- ***Radio and Antenna***

IRIS module features the Atmel AT86RF230 radio transceiver for wireless communication. The Atmel AT86RF230 is an IEEE 802.15.4 compliant RF transceiver. The maximum radio transmission power of IRIS module is 3 dBm. The following table shows the IRIS module different transmission power (dBm) contrast with power consumption (mA) [73].

Transmission Mode		**Receive Mode**
TX Power	Power Consumption	Power Consumption
3 dbm	17 mA	16 mA
-3 dbm	13 mA	
-17 dbm	10 mA	

Table 3-6 IRIS module power consumption in TX mode and RX mode

IRIS has a built-in antenna which provides 50 meters indoor range and up to 300 meters outdoor range. The antenna's performance also will be influenced by the battery pack [73].

3.2.2 Gateway MIB520

As shown in Figure 3-7, the base station also includes two parts: IRIS process/radio board and MIB520 USB gateway.

- ***Processor***

For the IRIS module comes with a low-power ATmega 1281 processor, it provides 8kb RAM, 128kb program flash memory, 512kb measurement flash. The ATmega

1281 processor only consumes 8mA and 8μA in the active mode and sleep mode, respectively [73]. In terms of MIB520 gateway, it has own on-board processor which is used to program the IRIS module. Actually, MIB 520 also supports MICA2 and MICAz platform. So any IRIS, MICA2, or MICAz module can be the base station when attached with MIB520 USB gateway.

- ***I/O Interfaces***

IRIS module supports a wide range of interfaces by 51-pin expansion connector, including UART, SPI, Digital I/O, I2C, and Analog inputs [73]. MIB520 supports USB interface which provides 57.6k baud rate and also supports 51-pin expansion connector for connecting with sensor node, and JTAG interface.

- ***Power and Voltage***

The IRIS module is powered by two AA batteries. The external power voltage is between 2.7V – 3.3V. MIB520 is supplied by USB bus power, so it does not need the external power source. However, once the MIB520 base station USB bus power is on, the AA battery power has to be switched off, otherwise, it may damage hardware.

- ***Radio and Antenna***

IRIS module features the Atmel AT86RF230 radio transceiver for wireless communication. The Atmel AT86RF230 is an IEEE 802.15.4 compliant RF transceiver.

CHAPTER 4. TEST-BED EXPERIMENTS

After viewing the previous chapter, the research related questions arise. What is the reliability of Castalia? How am I going to test it? To answer these questions, I walk through the following simple steps:

- Record the metrics of the IRISI test-bed on several scenarios.
- Simulate the same scenarios and collect the metrics from Castalia
- Compare the results of IRIS test-bed with the results collected from the simulation experiments.

This chapter briefly introduces the test-bed setup, test-bed experiments design principle, how the sensor nodes are programmed, and the experiment scenarios.

4.1 Test-Bed Setup

In this section, the process of setting up the Crossbow WSN starter kit will be discussed with highlighting some key points needed to pay attention. The test bed setup can be summarized in the following four parts:

1. Install MoteView software on the experiment PC.
2. Install the drivers for MIB520 USB base station
3. Setup IRIS sensor nodes
4. Start MoteView to configure the settings

[1]. Install MoteView software on the experiment PC.

There are a few requirements needed to check before the installation. Make sure PC operating system is Windows XP Home/ Professional edition or Windows 2000 with SP4. The PC has to use NTFS file system and at least 800 X 600 screen resolution. In addition, the most important step is that the administrator privilege account is needed to succeed in the installation. In this thesis, the windows XP professional OS was installed on the VMware virtual

machine.

[2]. Install the drivers for MIB520 USB base station

Connect the MIB520 USB base station to the available usb port on the PC. Then, windows detect the hardware and a pop up window will come out shows “Found new hardware”, asking to install the driver. So you need to choose “Install from a list or specific location”, then browse to the USB Drivers folder of CROSSBOW CD. Follow the instruction to finish the rest of driver installation. After driver installed, you will see that there are two virtual USB serial ports Computer Management ->Device Manager ->Ports (COM & LPT). From the below figure, we can see the two virtual serial ports are COM3 and COM4 in the experiment environment. COM3 is node programming, and COM4 is for node communication.

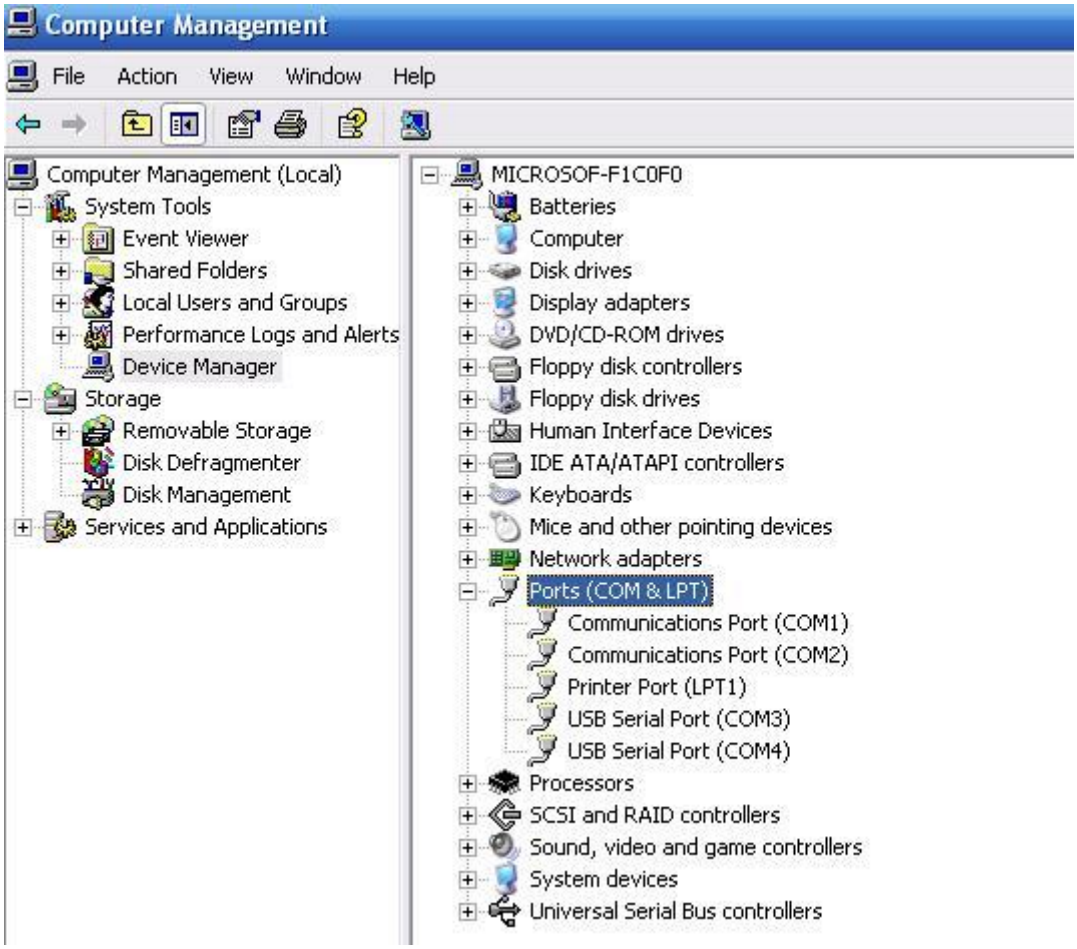

Figure 4-1 Device Manager

[3]. Setup IRIS sensor nodes

Insert AA batteries into battery stack of the sensor node and turn the power switch on. Put the sensor nodes on the table beside the base station.

[4]. Start MoteView to configure the settings

Start MoteView software, click “Connect to WSN” icon on the top left of the panel. The following Figure 4-2 is the pop-up window, in the Mode tap, select acquire live data as operation mode, and because of I installed the PostgreSQL database in the local windows XP machine, so I select local as acquisition type.

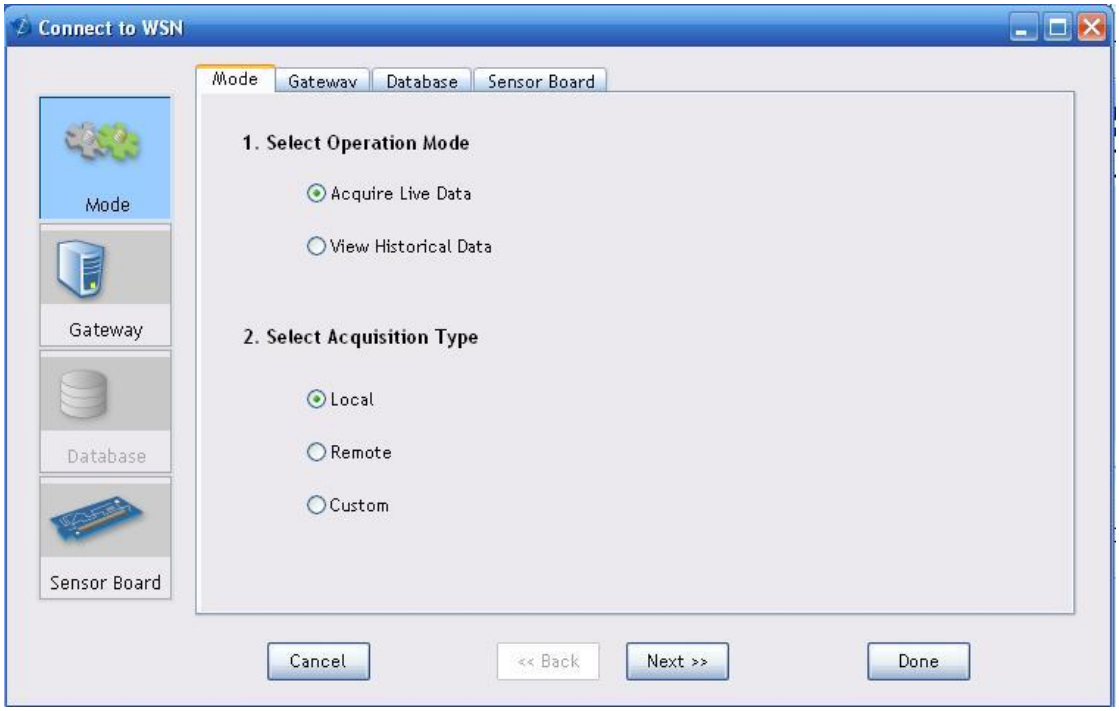

Figure 4-2 Mode Configurations

In the Gateway tab (Figure 4-3), select MIB520 from interface board, COM4 as serial port, and select 57600 as baud rate.

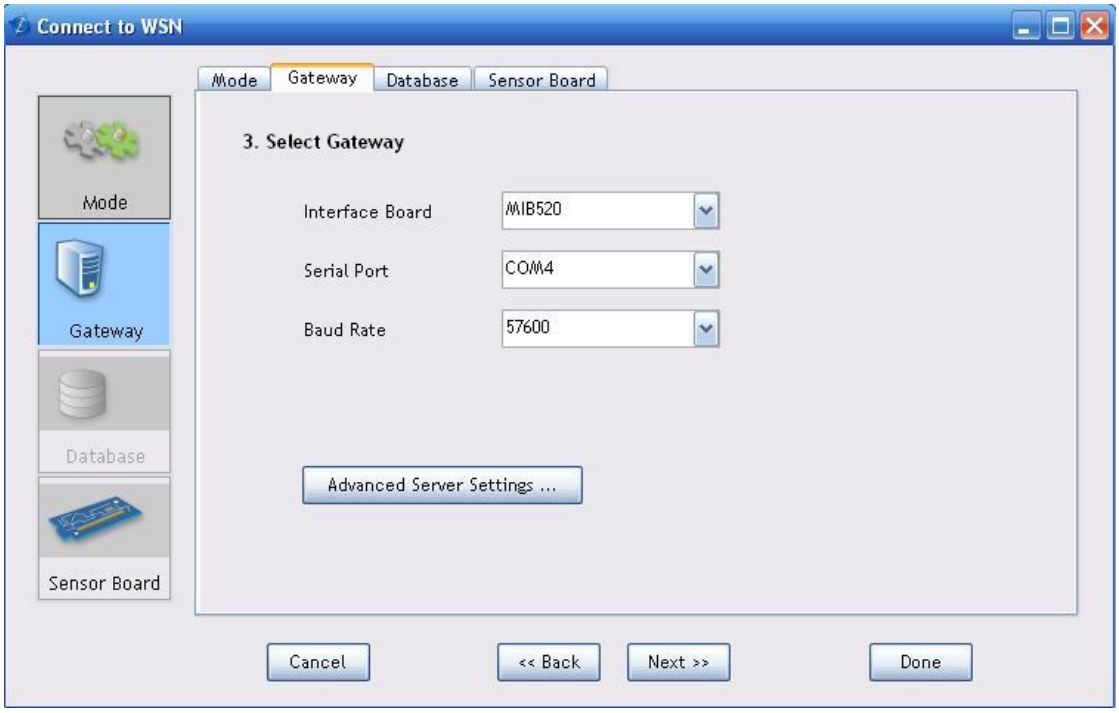

Figure 4-3 Gateway Configurations

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

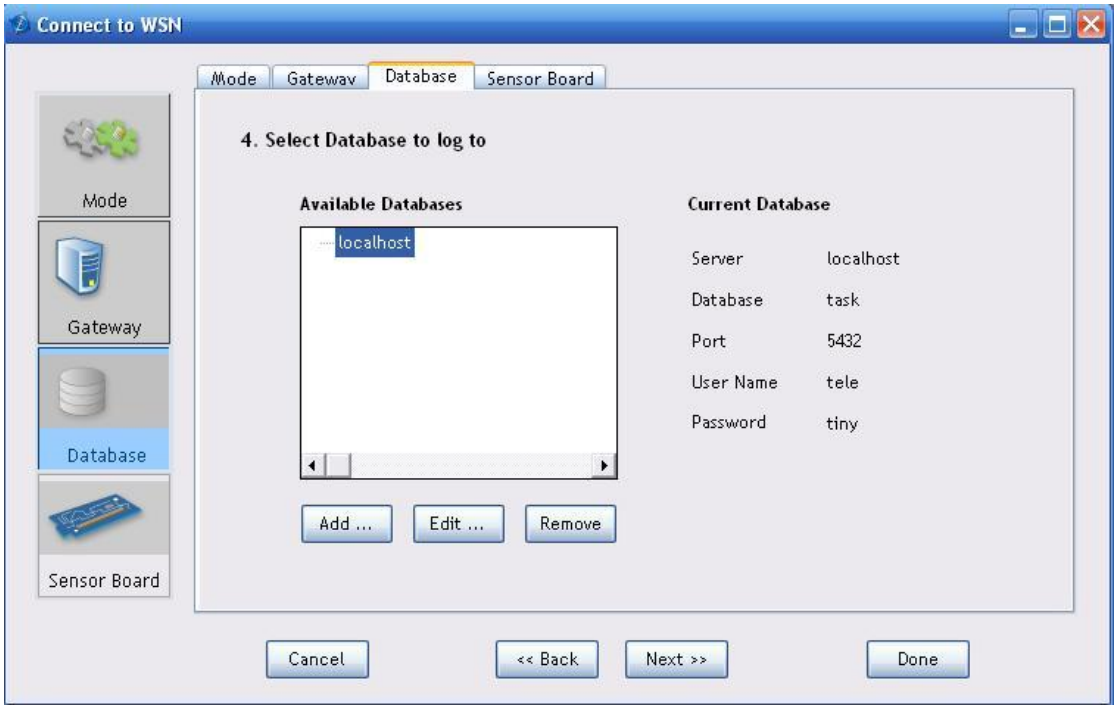

Figure 4-4 Database Configurations

In the Sensor Board tab (Figure 4-5), select XMTS400 from the application name drop list. Make sure View Alternate Table checkbox has been

unchecked. Then, click “Done”, you can see that IRIS nodes are flashing and MIB520 base station start to collect data from sensor nodes shows in Figure 4-6.

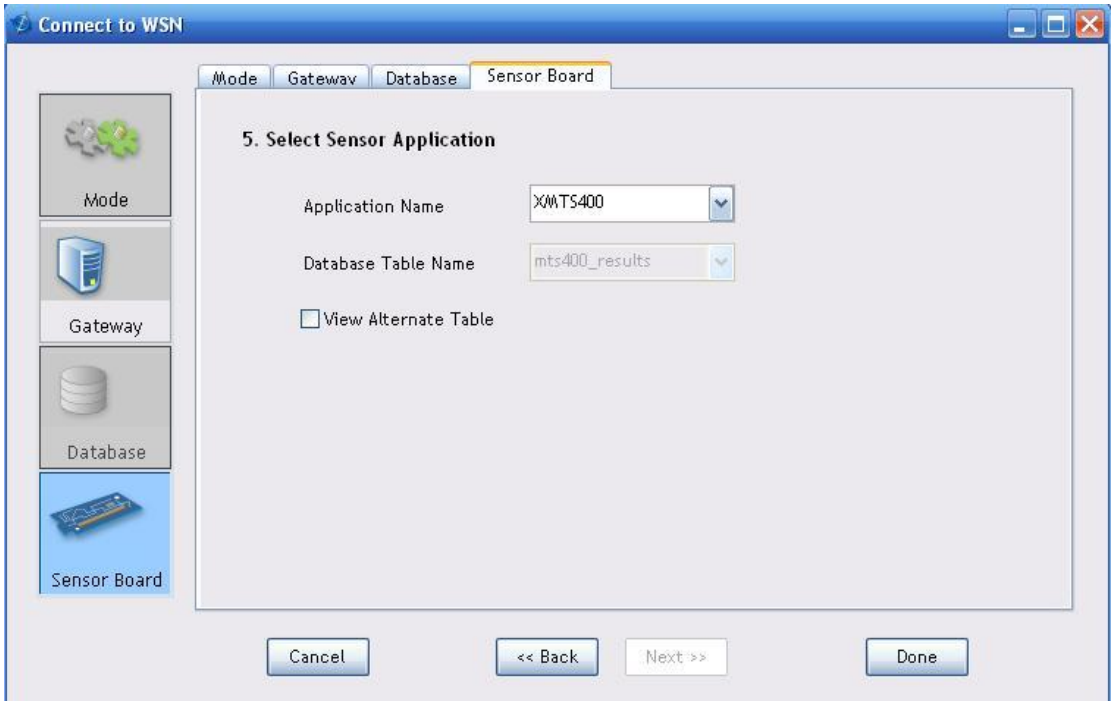

Figure 4-5 Sensor Board Configurations

Figure 4-6: IRIS Nodes & Gateway Setup

4.2 Design Principle

My test-bed experiments design principle follows as simple, accurate, and clear as possible. There are two reasons make us to decide this principle: firstly, I do not want the results comparison to be influenced by complicated technical issues. Because if there is little distance between IRIS WSN test-bed and simulation results in a simple and clear environment, the difference will be further expanded in a more complex environment. Secondly, resource constraints – we only have one pack Crossbow starter kit hardware as test bed which I've depicted earlier in this chapter.

4.3 Program Sensor Nodes

In terms of testing the reliability and accuracy of Castalia simulator, both real test-bed and the simulation will be set with the same initial parameters. This section describes how to program IRIS sensor nodes with initial parameters. Figure 4-6 shows the node 1 has been set to group id 125, packet size is 55 bytes, payload size is 48 bytes, and RF power is 0. The default IRIS node radio band is 2420 MHz.

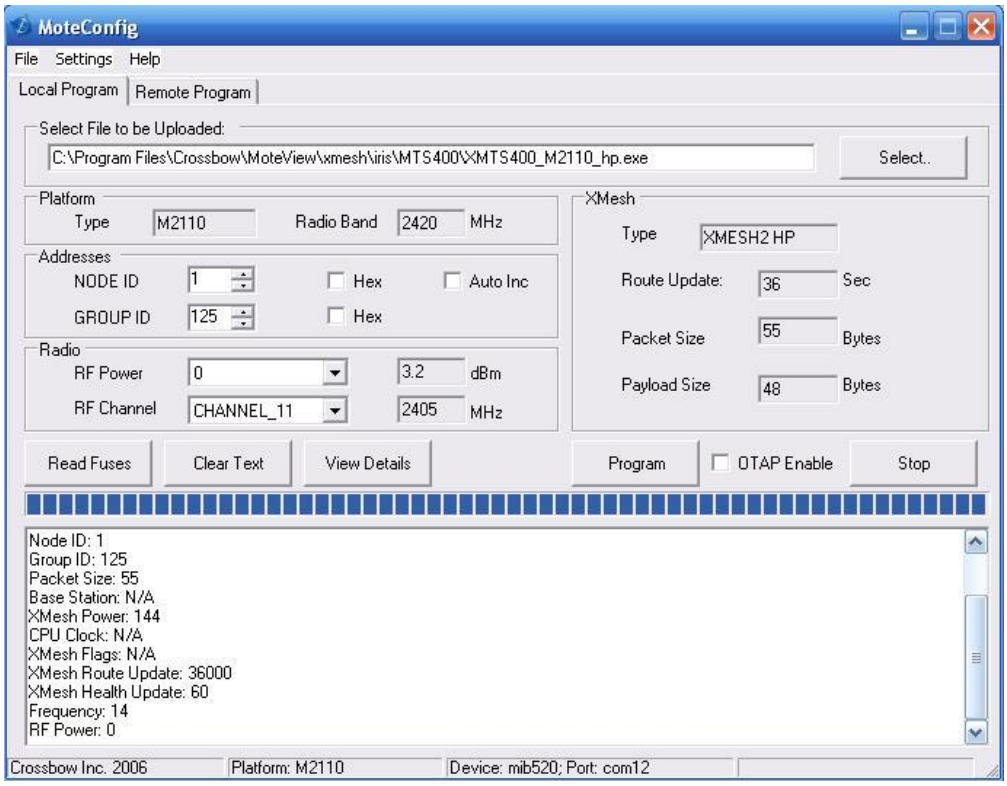

Figure 4-7: IRIS Node Programming

4.4 Experiments

In this thesis, there are two main experiments and a few scenarios completed by IRIS test-bed:

[1]. BridgeHealth (Static)

[2]. Robot Test (Mobility)

4.4.1 Bridge Health

This experiment was created to simulate the basic part of the wireless sensor network used for monitoring bridge structure health, deploying outdoor to minimize the effects of interferences and reflections as shown in the following figure 4-8.

There are a few scenarios in this static experiment:

A. Sink node (MIB520) and the sensor nodes are placed in a line and the

sensor nodes are on the same side as shown in Figure 4-8. However, there are the following sub-scenarios in the scenario A:

a) 2m x 5m – node 1 has 2m distance to sink node and node 2 has 5m distance to sink node.
b) 5m x 10m – node 1 has 5m distance to sink node and node 2 has 10m distance to sink node.

B. The sink node always goes at the center and the two sensor nodes are placed on the each side of the sink node. The following are the sub-scenarios of scenario B:

a) 2m x 5m – node 1 has 2m distance to sink node and node 2 has 5m distance to sink node.
b) 5m x 10m – node 1 has 5m distance to sink node and node 2 has 10m distance to sink node.

C. Repeat the scenario A and scenario B experiments by re-programming sensor nodes initial parameters, e.g., RF power, packet size, payload size, etc.

Figure 4-8: Bridge Health Experiment

4.4.2 Robot Test

In section 4.4.1 bridge health experiment, the sink node collects data from two static nodes, because the two IRIS nodes were static without mobility. But this robot experiment was designed to collect data from a mobile sensor node as shown in Figure 4-9.

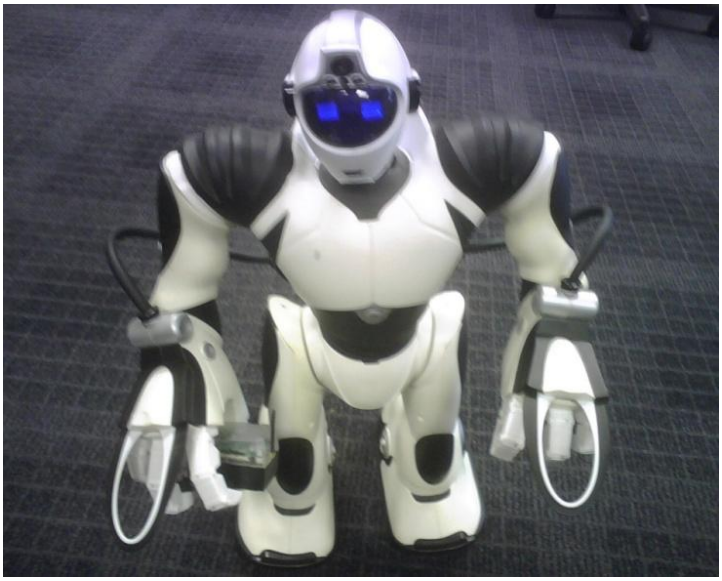

Figure 4-9: Robot Experiment

CHAPTER5.WSN SIMULATIONS

5.1 Simulation Settings

The Castalia simulations are performed on a mesh topology which is composed of a coordinator and two sensor nodes. All sensor nodes will report data to the coordinator (sink node). The default size of the data packet size and the payload size are 55 and 48 bytes respectively. CSMA/CA scheme is used to avoid collision in the simulation. The 2.4 GHz frequency band of CC2420 radio model is used and the PAN works in beacon-enabled mode. Each simulation lasts 120 seconds and each result is obtained from the average of 50 simulation results. The type of traffic loads are data packet rate and set to 5, 10, 20, 40, 60, 80 and 100 packets per second. As a result, the energy consumption of a sensor node includes receiving, transmitting, listening and sleeping states. The Castalia resource manager module simulates two AA batteries energy with 18720 Joules. All sensor nodes power consumption are subtracted based on overall power drown and the duration of simulation. There is only one routing protocol (Multipath Rings Protocol) considered in this thesis, because bypass routing protocol is seem as no routing implementation in Castalia version 3.1, the old routing protocols provided in Castalia version 2.3 were not realistic. For each of the two simulations, some of the same parameters were used especially those regarding the data packet rate, transmission power, interference level and throughput test model.

5.1.1 Simulation Environment

This simulation experiments were created to simulate the basic part of wireless sensor network used for monitoring bridge structure health. The simulation field follows the actual bridge size (field_x = 10m), but the field_y size is changed from

1.5 meters to 2 meters shown in Figure 4-8. The duration of simulation is 120 seconds same with the test-bed experiment time.

sim-time-limit = 120s

SN.field_x = 10 # meters

SN.field_y = 2 # meters

Radio transceiver module CC2420 was used in the two simulation experiments and all the scenarios. Because CC2420 chip has the most similar radio chip power consumption and specification with IRIS node radio transceiver Atmel AT86RF230. Figure 5-1 shows the Chipcon CC2420 and AT86RF230 radio transceiver parameter comparison.

	Radio Chip Power Consumption			Radio Chip Specification				
	Sleep	Idle/Rx	Tx [mA]	Frequency (MHz)	Modulation	Data Rate	Tx Power (dBm)	Rx Sensitivity (dBm)
TI CC2420	0.02-426uA	18.8mA	17.4(0 dBm)	2400-2483.5	OQPSK	250 kbps	-24 - 0	-95
Atmel AT86RF230	20nA	15.5mA	16.5(3 dBm)	2405-2480	OQPSK	250 kbps	-17 - 3	-101

Figure 5-1: CC2420 vs. AT86RF230 [52]

According to the test-bed experiment, the packet size and payload size are set to 55 bytes and 48 bytes in the Castalia simulation, respectively.

SN.node[*].Communication.Routing.maxNetFrameSize = 55

SN.node[*].Communication.MAC.maxMACFrameSize = 55

SN.node[*].Communication.Radio.maxPhyFrameSize = 55

SN.node[*].Application.constantDataPayload = 48

5.1.2 Simulation Scenarios

In order to keep the simulation data accuracy, the Castalia simulationscenarios exactly follow the test-bed experiments shown in section 4.4. But in this simulation more parameters will be considered and simulated, including various radio

transmission power, traffic load, beacon interval fraction, interference model, CCA threshold and GTS on/off.

5.2 Bridge Health Simulation (Static)

In the bridge health simulation, the nodes have no mobility. The table 5-1 Castalia code defines the sensor nodes location in scenario A& B:

Scenario A	Scenario B
SN.node[0].xCoor = 0	*SN.node[0].xCoor = 0*
SN.node[0].yCoor = 0	*SN.node[0].yCoor = 0*
SN.node[1].xCoor = 2	*SN.node[1].xCoor = 5*
SN.node[1].yCoor = 0	*SN.node[1].yCoor = 0*
SN.node[2].xCoor = 5	*SN.node[2].xCoor = 10*
SN.node[2].yCoor = 0	*SN.node[2].yCoor = 0*

Table 5-1: Sensor Nodes Initial Position

In the scenario A, Table 5-1 defines the initial location for sensor node 1 at (x=2, y=0) and sensor node 2 at (x=5, y=0). In the scenario B, Table 5-1 defines the initial location for sensor node 1 at (x=5, y=0) and sensor node 2 at (x=10, y=0).

The following are the vary parameters considered in the simulation scenarios:

- Transmission power and Packet rate - The vary radio transmission power and data packets rate are used in the simulation to see the difference how it affects the packet transmission results.The below packet rate means how many packets per second will be transmitted.

  SN.node[*].Communication.Radio.TxOutputPower = ${TxPower="0dBm", "-5dBm", "-10dBm", "-15dBm"}

  SN.node[*].Application.packet_rate = ${rate=5, 10, 20, 40, 60, 80, 100}
- Interference Model –this parameter depicts the radio collision model, there

are three levels of interferences: level 0, 1, and 2. in level 0, it assumes there is no collision at all. In level 1, it considers the collision happens if more than one node is sending data at the same time and the signal is over Signal Interference Ratio (SIR). In level 2, only the strongest signal is considered, all other signals are considered as noise.

- Configure vary MAC protocols include T-MAC, S-MAC,ZigBeeMAC, Carrier Sense Tunable MAC, Duty Cycle Tunable MAC. The below is the source codes for configuring each MAC protocol:

```
[Config TMAC]
SN.node[*].Communication.MACProtocolName = "TMAC"

[Config SMAC]
SN.node[*].Communication.MACProtocolName = "TMAC"
SN.node[*].Communication.MAC.listenTimeout = 61
SN.node[*].Communication.MAC.disableTAextension = true
SN.node[*].Communication.MAC.conservativeTA = false
SN.node[*].Communication.MAC.collisionResolution = 0

[ConfigZigBeeMAC]
SN.node[*].Communication.MACProtocolName = "Mac802154"
SN.node[0].Communication.MAC.isFFD = true
SN.node[0].Communication.MAC.isPANCoordinator = true
SN.node[*].Communication.MAC.phyDataRate = 1024
SN.node[*].Communication.MAC.phyBitsPerSymbol = 2

[ConfigjustCarrierSenseMAC]
SN.node[*].Communication.MACProtocolName = "TunableMAC"

[ConfigvaryDutyCycleMAC]
SN.node[*].Communication.MACProtocolName = "TunableMAC"
```

SN.node[*].Communication.MAC.dutyCycle={dutyCycle=0.03,0.05,0.1,0.3}

- Vary CCA Threshold – set the channel clear assessment parameter.
  SN.node[0].Communication.Radio.CCAthreshold = ${CCAthreshold=-95, -90, -85}
- Vary Beacon Interval Fraction
  SN.node[*].Communication.MAC.beaconIntervalFraction = ${beaconFraction= 0.2, 0.5, 0.8}
- *Vary Sigma levels*
  SN.wirelessChannel.sigma = ${Sigma=0,1,3,5}

5.3 Robot Test (Mobility)

The sensor node has mobility in this simulation experiment. The functionality is provided by Castalia mobility manager module. In order to test the implementation of sensor nodes mobility for Castalia simulator, two simple scenarios were designed to compare the results of simulations run on Castalia extension with those obtained for the same scenario when run on IRIS test-bed. The first simulation scenario involved placing two node in a line, sink node at position (0, 0), node 1 at position (2, 0) and node 2 at position (10, 2). At the start of the simulation, node 2 began sending packets to node 0 and to move at 0.2 m/s, with node 2 moving toward position (0, 2).

The below source code shows that node 2 was granted mobility.

SN.node[0].MobilityManagerName = "NoMobilityManager"

SN.node[1].MobilityManagerName = "NoMobilityManager"

SN.node[2].MobilityManagerName = "LineMobilityManager"

SN.node[2].xCoor = 10

SN.node[2].yCoor = 2

SN.node[2].MobilityManager.updateInterval = 100

SN.node[2].MobilityManager.xCoorDestination = 0

SN.node[2].MobilityManager.yCoorDestination = 2

SN.node[2].MobilityManager.speed = 0.2

CHAPTER 6. RESULTS AND COMPARATIVE ANALYSIS

6.1 Performance Measures

In this chapter, the results collected from test-bed experiments are compared with the results collected from the simulation experiments of the same scenarios on Castalia. Metrics are recorded at the end of each simulation experiment for each node. The following six traditional metrics for wireless sensor network are investigated in the simulations:

- Packet successful rate – it is defined as the total received packets over the total transmitted packets.
- Packet dropped rate – the dropped packet rate is defined as the total lost packets over the total transmitted packets.
- Energy consumption – it means the average energy consumption among sensor nodes.
- Traffic Load - The type of traffic loads are data packet rate and set to 5, 10, 20, 40, 60, 80 and 100 packets per second.
- Throughput – the throughput is defined as the total number of bits received per second at the sink node.
- Packet retry rate – the packet retry rate is identified as the total re-transmitted packets over the total transmitted packets.

6.2 Experiment Results

6.2.1 Packet Successful Rate

Figures 6-1 and 6-2 show the simulation results of the packet success rate of test-bed and Castalia simulation in bridge health monitoring environment (2m x 5m). The results show that the best success rate of test-bed is 99.10% in the five rounds experiment. The worst success rate is 90.30% in the test-bed experiment. Compared with Castalia simulation results, it seems that Castalia show similar performance in interference model 0 (ideal communication) and interference model 2 (which is the default value and considered as realistic communication. All of the three kinds of interference models had been tested with over thousand test-bed and simulation experiments by the Castalia developers). The worst success rate of Castalia simulation is 44.12% in interference model 1, so that was the expected result because interference model 1 has more collisions than reality.In this scenario, the test-bed sensor node behaviour is very similar to the “IDEAL” communication in the Castalia, where all the packets are correctly transmitted and received, that neglects any complex aspect of the wireless channel such propagation, interferences and collisions, produces the best results.

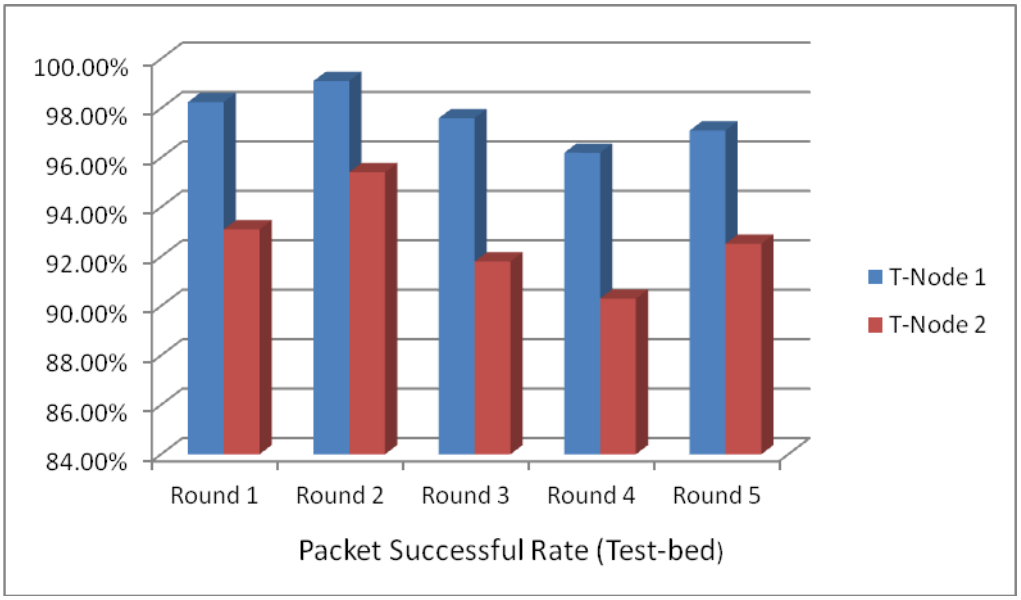

Figure 6-1: Test-bed Packet Successful Rate (2m x 5m)

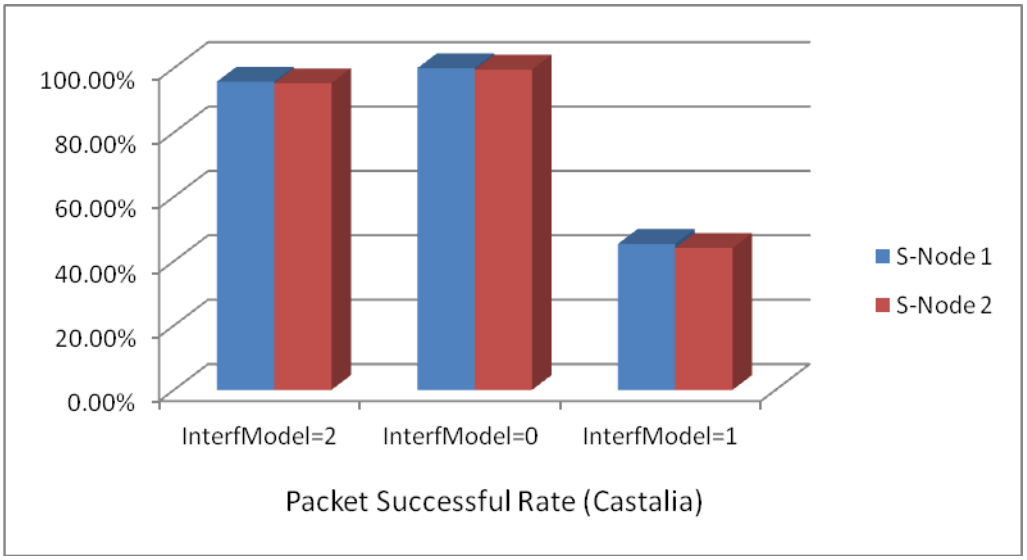

Figure 6-2: Castalia Packet Successful Rate (2m x 5m)

Table 6-1 shows the deviation of success rate between test-bed and Castalia simulation in interference model 0 and 2. The biggest deviation was 6.08%, happened on node 2 in interference model 0.

	Test-Bed	Simulation	Deviation
	Average	Collision=0	
Node 1	98.32%	100%	1.68%
Node 2	93.44%	99.52%	6.08%
	Average	Collision=2	
Node 1	98.32%	95.81%	2.51%
Node 2	93.44%	95.32%	-1.88%

Table 6-1: The deviation of packet successful rate in Castaliainterfmodel 0 and 2

Based on graph presented in Figures 6-3 and 6-4, they show that the simulation results of the packet success rate of test-bed and Castalia simulation in bridge health monitoring environment (5m x 10m). The results show that the best success rate of test-bed is 99.52% in the five rounds experiment.The worst success rate is

92.33% in the test-bed experiment. The 5m x 10m experiment results were even better than the results in 2m x 5m experiment. Compared with Castalia simulation results, it seems that Castalia still show similar performance in interference model 0and interference model 2. The worst success rate of Castalia simulation is 42.68% in interference model 1, so that was the expected result because interference model 1 has more collisions than reality.In the 5m x 10m experiment and simulation, the deviation of packet successful rate has not been extended. The biggest deviation was even reduced to 5.32%, happened on node 2 in interference model 0.

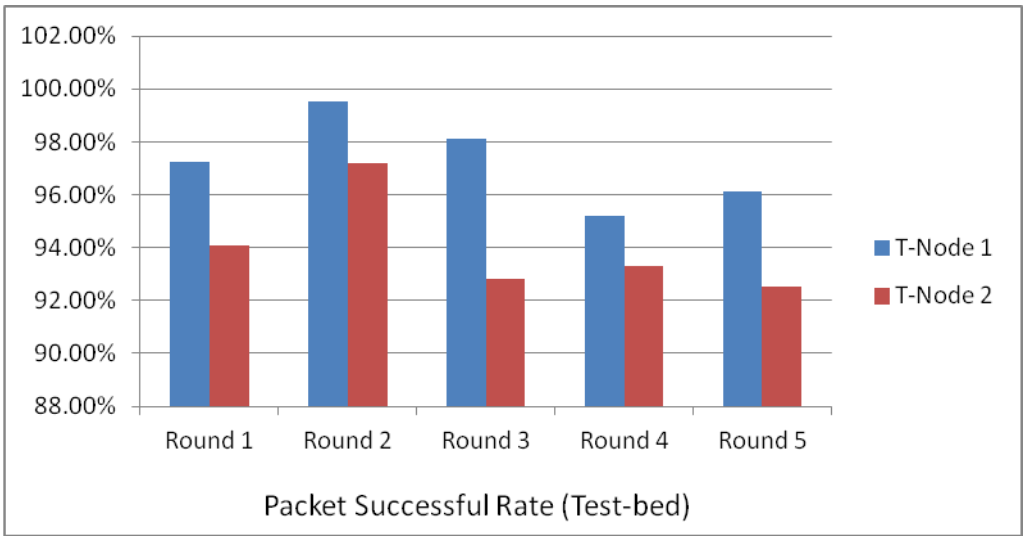

Figure 6-3: Test-bed Packet Successful Rate (5m x10m)

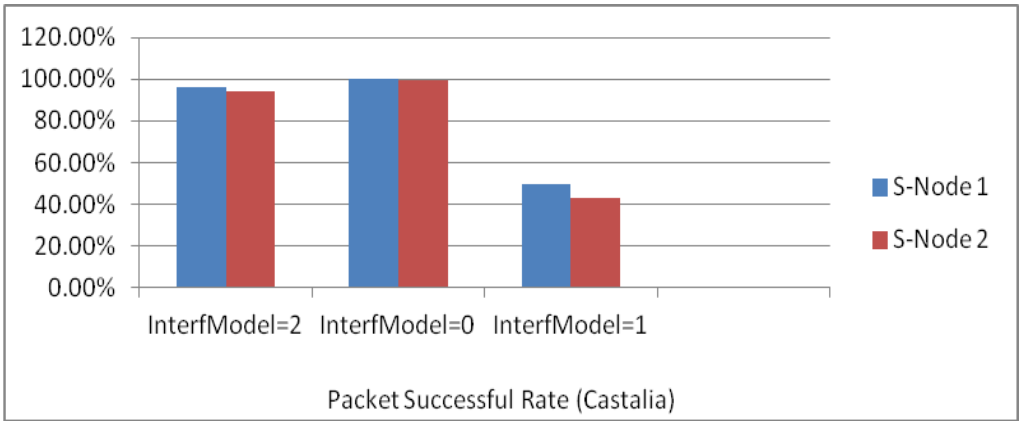

Figure 6-4: Castalia Packet Successful Rate (5m x10m)

For the traffic load, the success rate of test-bed and Castalia both decreases as the traffic load increase shown in figures 6-5 and 6-6. In the simulation scenario A, the worst success rates of test-bed and Castalia are 64.25 and 50.76% when the traffic load is 100 packets per second. The deviation of success rate between test-bed and Castalia simulation in various traffic loads from 2.1% to 13.49%. The reasons cause the results distance between simulation and test-bed is that the simulation default setting uses low RSSI and considers GTS (Guaranteed Time Slot) in this scenario.

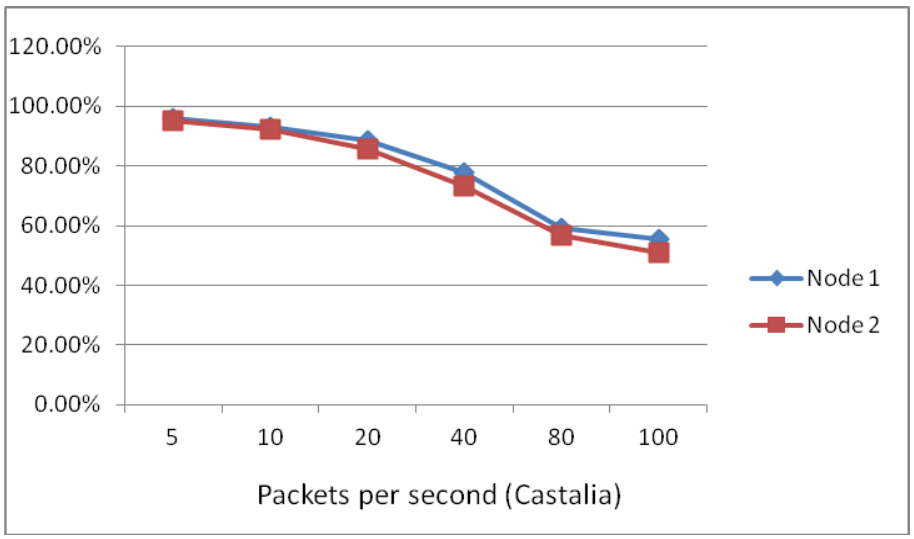

Figure 6-5: Castalia Packet Successful Rate with various traffic loads (2m x 5m)

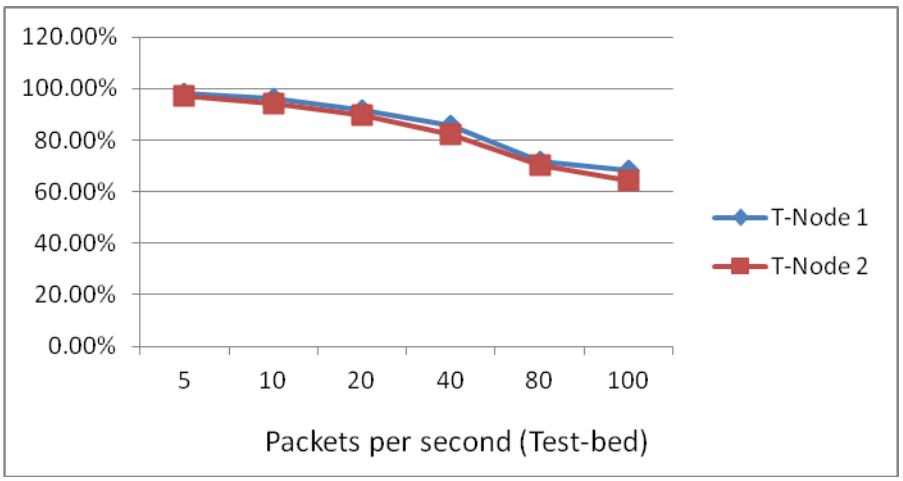

Figure 6-6: Test-bed Packet Successful Rate with various traffic loads (2m x 5m)

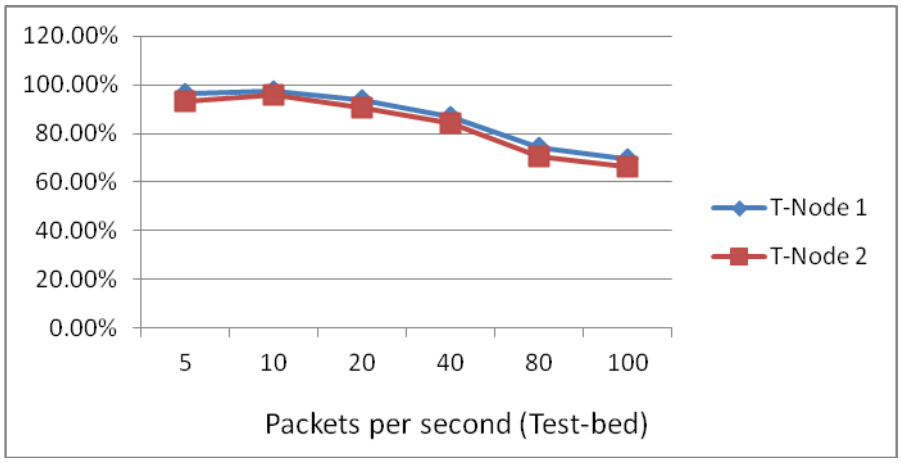

Figure 6-7: Test-bed Packet Successful Rate with various traffic loads (5m x 10m)

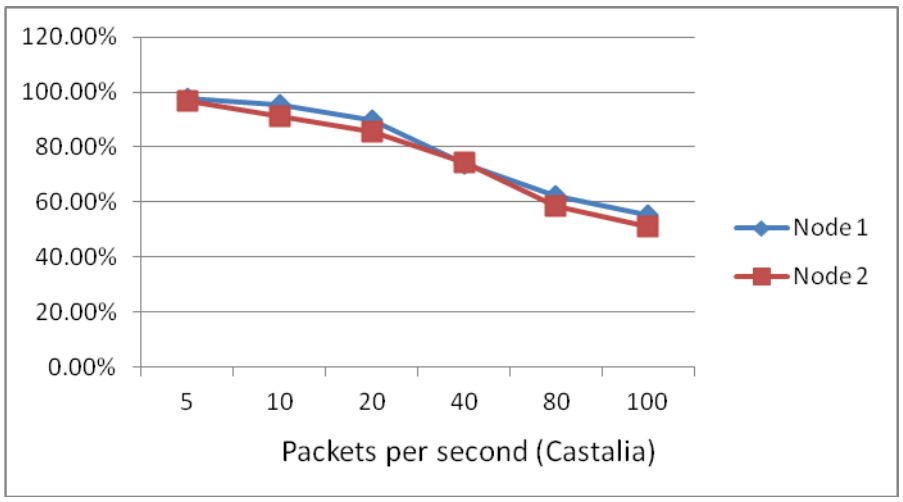

Figure 6-8: Castalia Packet Successful Rate with various traffic loads (5m x 10m)

Based on graph presented in Figures 6-7 and 6-8, they show that the success rate of test-bed and Castalia both decreases as the traffic load increase in simulation scenario B (5m x 10m). In this simulation scenario, the worst success rates of test-bed and Castalia are 66.16 and 51.29% when the traffic load is 100 packets per second. The deviation of success rate between test-bed and Castalia simulation in various traffic loads from 3.39% to 14.96%.

6.2.2 Packet Retry Rate

Figures 6-9 and 6-10 show the packet retry rate which is the number of retry

packets in the total number of transmissions. The worst packets retry rates of test-bed and Castalia are 1.51% and 12.15% in experiment scenario 2.

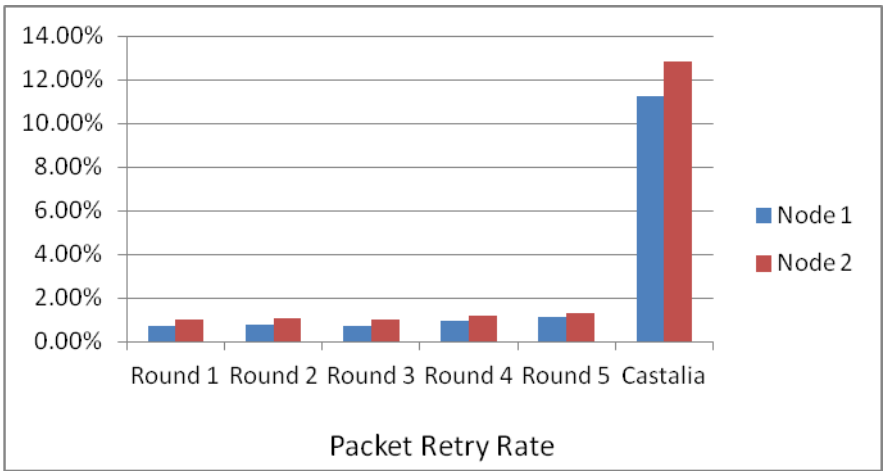

Figure 6-9: Test-bed vs. Castalia Packet Retry Rate (2m x 5m)

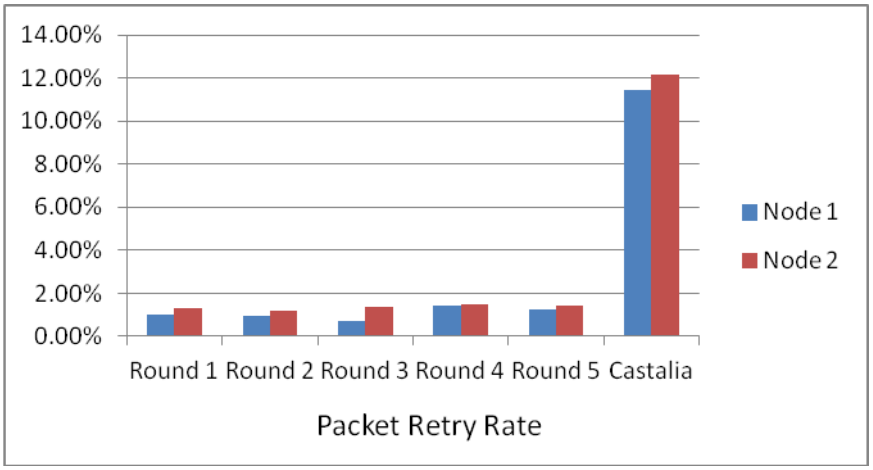

Figure 6-10: Test-bed vs. Castalia Packet Retry Rate (5m x 10m)

Table 6-2 shows the deviation of packet retry rate between test-bed average result and Castalia simulation scenario A. The deviation for node1 and node2 were 10.4% and 11.71% respectively. In the experiment scenario B, the deviation for node1 and node2 were 10.36% and 10.8% respectively shown in table 6-3.

	Test-Bed	Simulation	Deviation
	Average	Average	
Node 1	0.87%	11.27%	10.4%
Node 2	1.12%	12.83%	11.71%

Table 6-2: The deviation of packet retry rate (2m x 5m)

	Test-Bed	Simulation	Deviation
	Average	Average	
Node 1	1.06%	11.42%	10.36%
Node 2	1.35%	12.15%	10.8%

Table 6-3: The deviation of packet retry rate (5m x 10m)

6.2.3 Packet Dropped Rate

Figures 6-11 and 6-12 show the packet dropped rate which is the number of dropped packets in the total number of transmissions.The worst packet dropped rates of test-bed and Castalia are 1.47% and 4.36%. The simulation scenario used the realistic communication in interference model 2 and the traffic load is 5 packets per second.

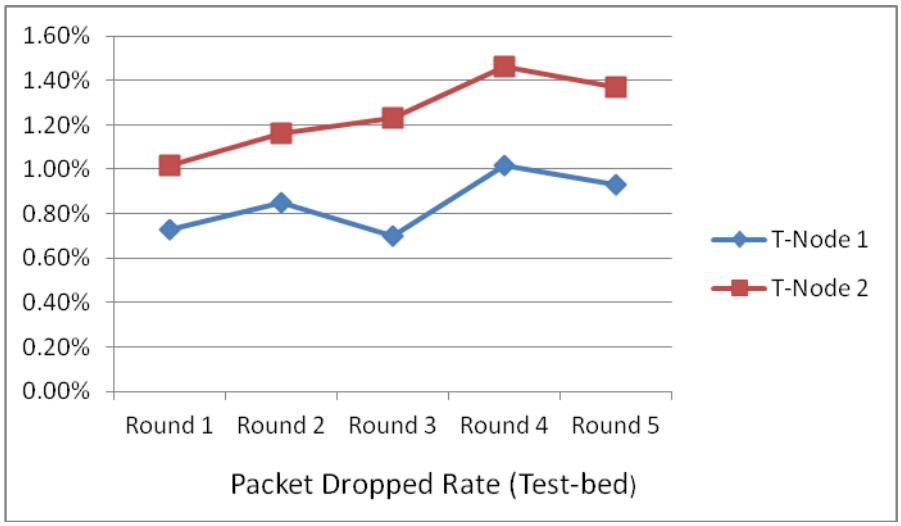

Figure 6-11: Test-bed Packet Dropped Rate (2m x 5m)

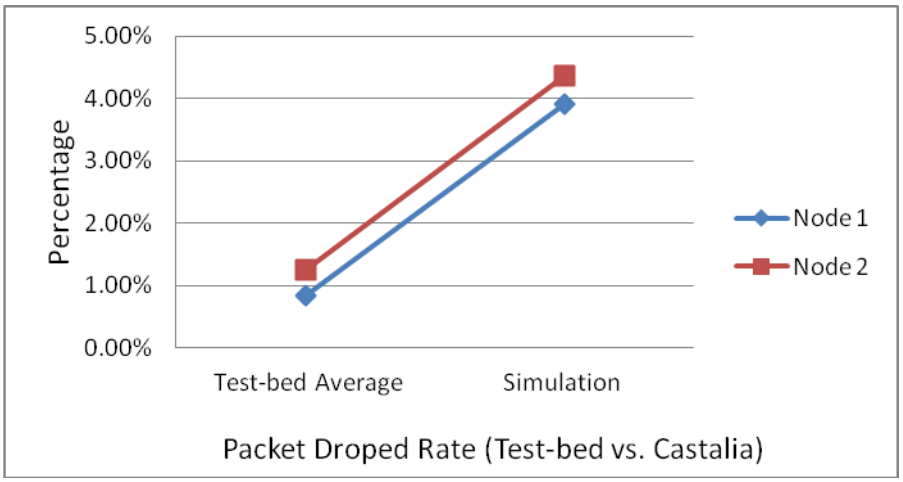

Figure 6-12: Test-bed vs. Castalia Packet Dropped Rate (2m x 5m)

Table 6-4 shows the deviation of packet dropped rate between test-bed average result and Castalia simulation scenario A in interference model 2. The deviation for node1 and node2 were 3.07% and 3.11% respectively.

	Test-Bed	Simulation	Deviation
	Average	Collision=2	
Node 1	0.85%	3.92%	3.07%
Node 2	1.25%	4.36%	3.11%

Table 6-4: The deviation of packet dropped rate (2m x 5m)

In the simulation scenario B (5m x 10m), there is almost not different in the results compared with simulation scenario A. Table 6-3 the deviation of packet dropped rate between test-bed average result and Castalia simulation scenario B. The deviation for node1 and node2 were 4.35% and 4.75% respectively.

	Test-Bed	Simulation	Deviation
	Average	Collision=2	
Node 1	0.97%	5.32%	4.35%
Node 2	1.41%	6.16%	4.75%

Table 6-5: The deviation of packet dropped rate (5m x 10m)

Figure 6-13 shows the packet dropped rate of test-bed and Castalia both increases as the traffic load increases. The worst packet dropped rates of test-bed and Castalia are 23.37 and 41.93% when the traffic load is 100 packets per second. The deviation of dropped rate between test-bed and Castalia simulation in various traffic loads from 3.01% to 18.56%. The GTS off and the low RSSI value are the two main reasons extended the results distance between Castalia simulation and test-bed.

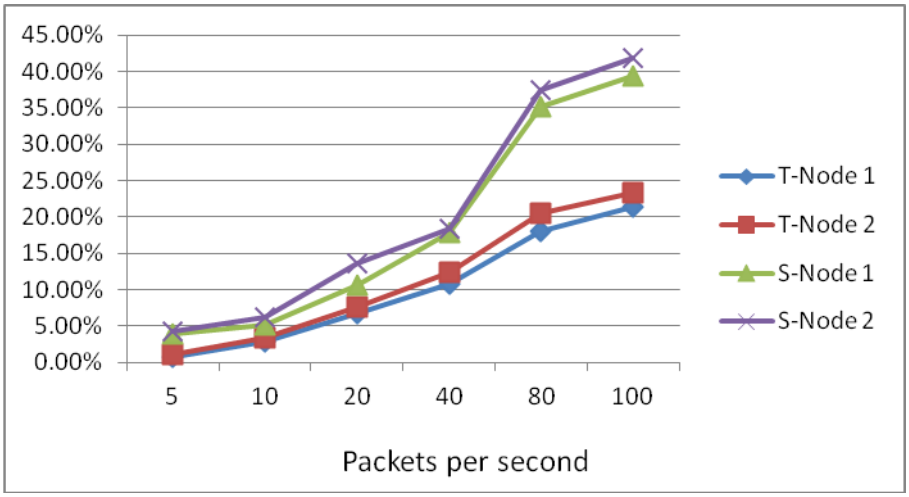

Figure 6-13: Test-bed vs. Castalia Packet Dropped Rate with various traffic loads

Based on the graph presented in Figure 6-14, it shows that the packets were received per node with various packet rates in Castalia simulation. The x axis is the sending rate for each node measured in packets/second. Generally the simulation performs better when the GTS is turned on. This is something to be expected as TDMA schemes make a more efficient use of the wireless medium and are reducing interference.

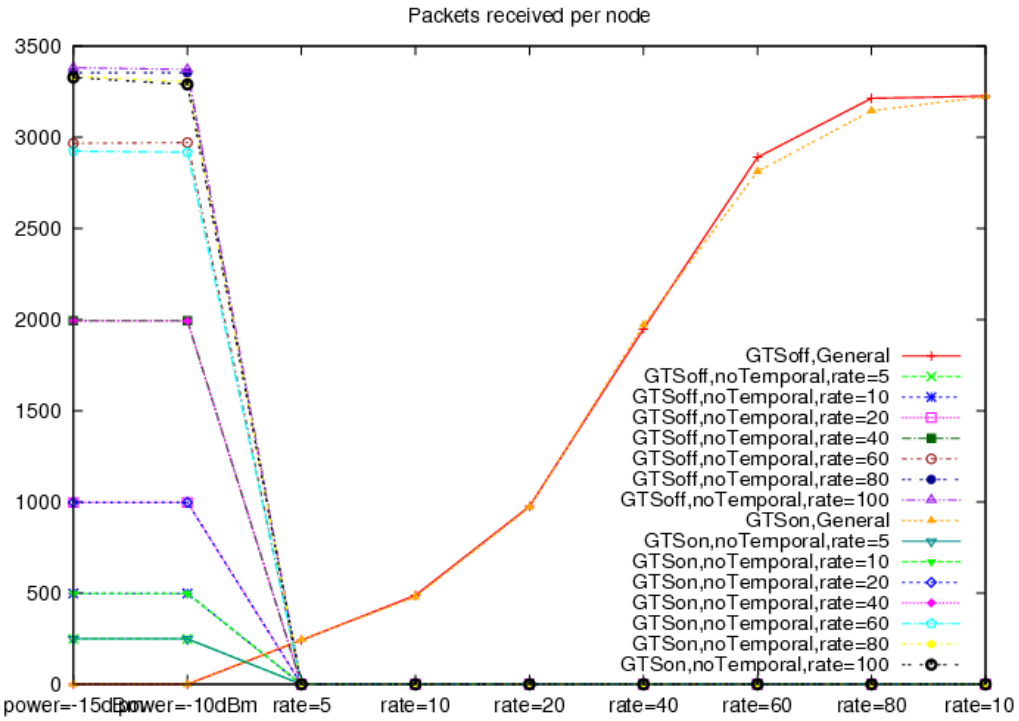

Figure 6-14: Castalia Packet received per node with various packet rates

6.2.4 Energy Consumption

In the following, the energy consumption of test-bed and Castalia simulations are compared. The energy consumption of each node is the summation of energy consumptions of a node in transmission, receiving, listening and sleeping modes. However, the test-bed experiment does not provide reliable results. It may because of the short simulation duration which does not consume much energy. Therefore, the test-bed ignores the energy consumption. Figure 6-15 shows the energy consumption of Castalia increases as the traffic load increases. Figure 6-16 shows the energy consumption of Castalia decreases as the transmission power decreases.

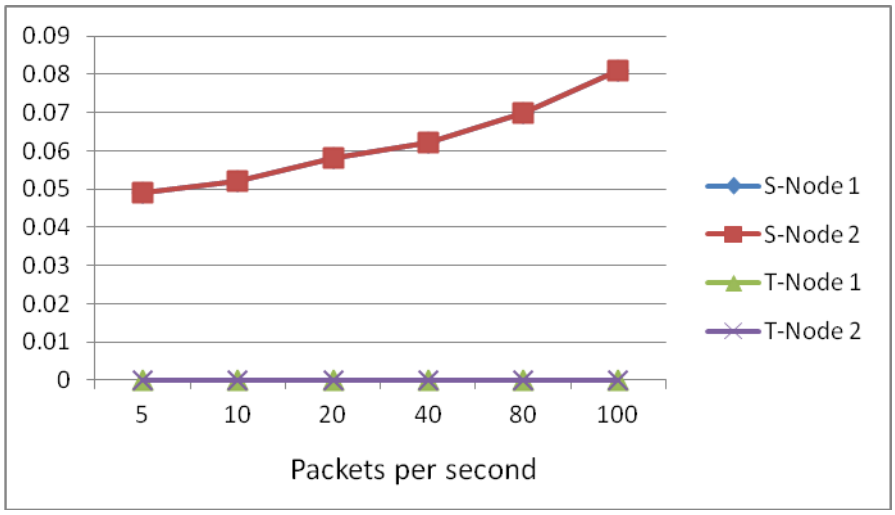

Figure 6-15: Castalia vs. Test-bed Energy Consumption

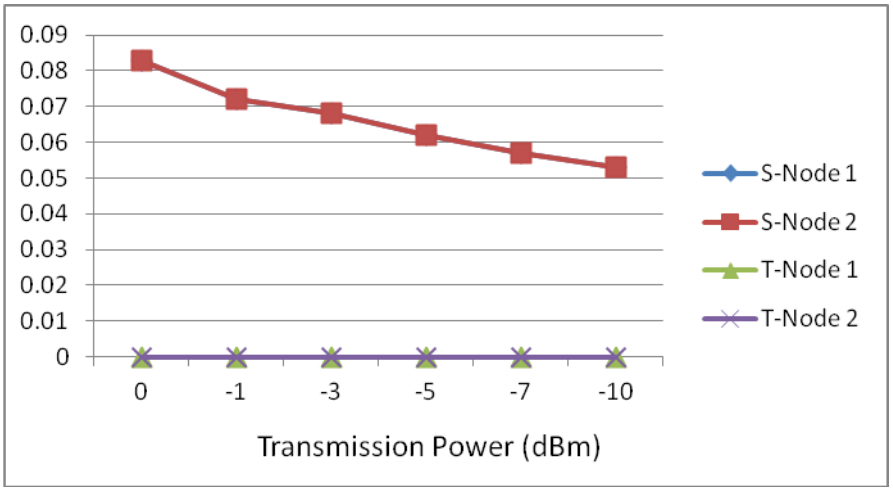

Figure 6-16: Castalia vs. Test-bed Energy Consumption

CHAPTER 7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

This thesis presented a simulator known as Castalia which is based on OMNET++ platform, for simulating wireless sensor network on a small to large scale network. So far, the results gathered from an experiment conducted on a real test-bed and the results from simulation had been compared and analyzed, in order to evaluate the reliability and the scalability of the Castalia simulator. In this study, various traffic loads, radio propagation, transmission power, packet rates, MAC frame sizes, interference, and simulation time in the simulator were also implemented. The experiments showed that Castalia provided quite reliable results on the considered scenarios as it outperformed the results from the experiments conducted on the test-bed by 2% - 20%. In addition, a simple tuning Castalia default value configurations can significantly increase the accuracy. Nevertheless, Castalia shows slightly a better performance in terms of the reliability of the simulation results and substantially better results in terms of efficiency. Recall, Castalia interferences are dynamically calculated [41].The guess was that this calculation is somehow inaccurate. In conclusion, Castalia is a reliable simulator. It is well designed and allows researchers and developers to easily implement and validate their own algorithms and protocols for WSN.

7.2 Future Work

There are several improvements that can be done to this study in the future:

- Find out the reasons of poor behaviour of Castalia in the considered scenarios.
- With the new release of Castalia 3.2, start by applying the same approach taken in this thesis and then implement it.

- Investigate the reliability of other frameworks for WSN modeling in OMNET++, such as NS-2 (32, 33).
- To integrate some other routing protocols to this simulation.
- Chocse which parameters to be measured by the sensor nodes and reprogram the IRIS test-bed nodes with the various parameter value settings.

References

[1]. Kulakowski, P. (2008). WIRELESS SENSOR NETWORKS: TECHNOLOGY, PROTOCOLS, AND APPLICATIONS. [Book Review]. *IEEE Communications Magazine, 46*(6), 42-44.

[2]. Roy, P. L.-H. S. (2010). Low-Power Wake-Up Radio for Wireless Sensor Networks. *Mob. Netw. Appl., 15*(2), 226-236. doi: 10.1007/s11036-009-0184-3

[3]. Bohli, J.-M., Hessler, A., Ugus, O., &Westhoff, D. (2008). *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.

[4]. Colesanti, U. M., Crociani, C., &Vitaletti, A. (2007). *On the accuracy of omnet++ in the wireless sensornetworks domain: simulation vs. testbed*. Paper presented at the Proceedings of the 4th ACM workshop on Performance evaluation of wireless ad hoc, sensor,and ubiquitous networks, Chania, Crete Island, Greece.

[5]. Orfanus, D., Lessmann, J., Janacik, P., &Lachev, L. (2008). *Performance of wireless network simulators: a case study*. Paper presented at the Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, Vancouver, British Columbia, Canada.

[6]. Xu, N., Rangwala, S., Chintalapudi, K. K., Ganesan, D., Broad, A., Govindan, R., &Estrin, D. (2004). *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.

[7]. Hu, W., Bulusu, N., Chou, C. T., Jha, S., Taylor, A., & Tran, V. N. (2009). 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

[8]. P, R., & P, P. M. (2011). 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

[9]. Huynh, A., Jingcheng, Z., Qin-Zhong, Y., &Shaofang, G. (2010). ZigBee Radio with External Power Amplifier and Low-Noise Amplifier. [Article]. *Sensors & Transducers (1726-5479), 118*(7), 110-121.

[10]. Rasin, Z., & Abdullah, M. R. (2009). Water Quality Monitoring System Using Zigbee Based Wireless Sensor Network. [Article]. *International Journal of Engineering & Technology, 9*(10), 24-28.

[11]. Jingcheng, Z., Allan, H., Qinzhong, Y., &Shaofang, G. (2010). Design of the Remote Climate Control System for Cultural Buildings Utilizing ZigBee Technology. [Article]. *Sensors & Transducers (1726-5479), 118*(7), 13-27.

[12]. ZigBee. (2011). ZigBee Alliance Retrieved 15/Aug/2011, from http://www.zigbee.org

[13]. Titzer, B. L., Lee, D. K., &Palsberg, J. (2005). *Avrora: scalable sensor network simulation with precise timing*. Paper presented at the Proceedings of the 4th international symposium on Information processing in sensor networks, Los Angeles, California.

[14]. Alberola, R. d. P., &Pesch, D. (2008). *AvroraZ: extending Avrora with an IEEE 802.15.4 compliant radio chip model*. Paper presented at the Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, Vancouver, British Columbia, Canada.

[15]. NetTopo. (2008). NetTopo Retrieved 25/04/2011, from http://www.semanticreality.org/nettopo/index.htm

[16]. Shu, L., Wu, C., Zhang, Y., Chen, J., Wang, L., &Hauswirth, M. (2008). NetTopo: beyond simulator and visualizer for wireless sensor networks. *SIGBED Rev., 5*(3), 1-8. doi: 10.1145/1534490.1534492

[17]. Shu, L., Hauswirth, M., Wang, L., Yuan, Z., & Chen, Y. (2009). *Visualizing simulation and testbed of wireless sensor networks with NetTopo*. Paper presented at the Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization, Beijing, China.

[18]. NetTopo. (2011). NetTopo Retrieved 24/04/2011, from http://lei.shu.deri.googlepages.com/nettopo

[19]. Haiming, C., Li, C. U. I., Changcheng, H., & He, Z. H. U. (2009). EasiSim: A Scalable Simulator for Wireless Sensor Networks. [Article]. *Wireless Sensor Network, 1*(5), 467-474. doi: 10.4236/wsn.2009.15056

[20]. Levis, P., Lee, N., Welsh, M., & Culler, D. (2003). *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.

[21]. Gay, D., Levis, P., Behren, R. v., Welsh, M., Brewer, E., & Culler, D. (2003). *The nesC language: A holistic approach to networked embedded systems*. Paper presented at the Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, San Diego, California, USA.

[22]. Feng, C., Nan, W., German, R., & Dressler, F. (2010). Simulation study of IEEE 802.15.4 LR-WPAN for industrial applications. [Article]. *Wireless Communications & Mobile Computing, 10*(5), 609-621. doi: 10.1002/wcm.736

[23]. Rohm, D., Goyal, M., Hosseini, H., Divjak, A., & Bashir, Y. (2009). A simulation based analysis of the impact of IEEE 802.15.4 MAC parameters on the performance under different traffic loads. [Article]. *Mobile Information Systems, 5*(1), 81-99. doi: 10.3233/mis-2009-0074

[24]. Mahdi Alavi, S. M., Walsh, M. J., & Hayes, M. J. (2010). Robust power control for IEEE 802.15.4 wireless sensor networks with round-trip time-delay uncertainty. [Article]. *Wireless Communications & Mobile Computing, 10*(6), 811-825. doi: 10.1002/wcm.791

[25]. Jangra, A. (2010). Wireless Sensor Network (WSN): Architectural Design issues and Challenges. [Article]. *International Journal on Computer Science & Engineering, 2*(9), 3089-3094.

[26]. AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. 2002. A survey on wireless sensor networks.*IEEE Comm. Mag. 40*, 8,

102–114.

[27]. Egbogah, E. E., &Fapojuwo, A. O. (2011). A Survey of System Architecture Requirements for Health Care-Based Wireless Sensor Networks. [Article]. *Sensors (14248220), 11*(5), 4875-4898. doi: 10.3390/s110504875

[28]. Bhatti, S., Carlson, J., Dai, H., Deng, J., Rose, J., Sheth, A., . . . Han, R. (2005). MANTIS OS: an embedded multithreaded operating system for wireless micro sensor platforms. *Mob. Netw. Appl., 10*(4), 563-579. doi: 10.1145/1160162.1160178

[29]. Han, C.-C., Kumar, R., Shea, R., Kohler, E., &Srivastava, M. (2005). *A dynamic operating system for sensor nodes*. Paper presented at the Proceedings of the 3rd international conference on Mobile systems, applications, and services, Seattle, Washington.

[30]. Gu, L., &Stankovic, J. A. (2006). *T-kernel: providing reliable OS support to wireless sensor networks*. Paper presented at the Proceedings of the 4th international conference on embedded networked sensor systems, Boulder, Colorado, USA.

[31]. Loukas, A., Woehrle, M., &Langendoen, K. (2011). *On mining sensor network software repositories*. Paper presented at the Proceeding of the 2nd workshop on Software engineering for sensor network applications, Waikiki, Honolulu, HI, USA.

[32]. NS-2. (2011). The Network Simulator - ns-2 Retrieved 15/05/2011, from http://www.isi.edu/nsnam/ns/

[33]. Stevens, C., Lyons, C., Hendrych, R., Carbajo, R. S., Huggard, M., &Goldrick, C. M. (2009). *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.

[34]. Sundresh, S., Kim, W., & Agha, G. (2004). *SENS: A Sensor, Environment and Network Simulator*. Paper presented at the Proceedings of the 37th annual symposium on Simulation.

[35]. Opnet. (2011). Opnet Retrieved 26/04/2011, from http://www.opnet.com/

[36]. Park, S., Savvides, A., &Srivastava, M. B. (2000). *SensorSim: a simulation framework for sensor networks*. Paper presented at the Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, Boston, Massachusetts, United States.

[37]. Akkaya, K., &Younis, M. (2005). A survey on routing protocols for wireless sensor networks. [Article]. *Ad Hoc Networks, 3*(3), 325-349. doi: 10.1016/j.adhoc.2003.09.010

[38]. Yi, R., Oleshchuk, V., Y. Li, F., &Xiaohu, G. (2011). Security in Mobile Wireless Sensor Networks - A Survey. [Article]. *Journal of Communications, 6*(2), 128-142. doi: 10.4304/jcm.6.2.128-142

[39]. Olivares, T., Tirado, P. J., & Orozco-Barbosa, L. (2006). *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.

[40]. Vu, T. M., Williamson, C., &Safavi-Naini, R. (2009). *Simulation modeling of secure wireless sensor networks*. Paper presented at the Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools, Pisa, Italy.

[41]. Boulis, A. (2010). Castalia: A simulator for Wireless Sensor Networksand Body Area Networks Retrieved 20/Nov/2010, 2010, from http://castalia.npc.nicta.com.au/index.php

[42]. Park, S., Savvides, A., &Srivastava, M. B. (2001). *Simulating networks of wireless sensors*. Paper presented at the Proceedings of the 33nd conference on winter simulation, Arlington, Virginia.

[43]. Glaser, J., Weber, D., Madani, S. A., &Mahlknecht, S. (2008). Power Aware Simulation Framework for Wireless Sensor Networks and Nodes. [Article]. *EURASIP Journal on Embedded Systems*, 1-16. doi: 10.1155/2008/369178

[44]. Ali, Q. I., Abdulmaowjod, A., & Mohammed, H. M. (2011). Simulation &

Performance Study of Wireless Sensor Network (WSN) Using MATLAB. [Article]. *Iraqi Journal for Electrical & Electronic Engineering, 7*(1), 112-119.

[45]. Song, J., Han, S., Zhu, X., Mok, A. K., Chen, D., & Nixon, M. (2008). *A complete wirelessHART network*. Paper presented at the Proceedings of the 6th ACM conference on embedded network sensor systems, Raleigh, NC, USA.

[46]. Yang, Y. (2008). MiWi P2P wireless protocol Retrieved 20/06/2011, from http://www.eetindia.co.in/STATIC/PDF/200812/EEIOL_2008DEC04_RFD_AN_01.pdf?SOURCES=DOWNLOAD

[47]. Polley, J. B., D., McGee, J., Rusk, D.,Baras, J.S. (2004). *ATEMU: a fine-grained sensor network simulator* Paper presented at the Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference USA.

[48]. Boulis, A. (2007). *Castalia: revealing pitfalls in designing distributed algorithms in WSN*. Paper presented at the Proceedings of the 5th international conference on embedded networked sensor systems, Sydney, Australia.

[49]. NESL, U. (2008). SOS embeded operating system Retrieved 02/06/2011, from https://projects.nesl.ucla.edu/public/sos-2x/doc/

[50]. Chang, Y.-C., &Sheu, J.-P. (2009). An Energy Conservation MAC Protocol in Wireless Sensor Networks. [Article]. *Wireless Personal Communications, 48*(2), 261-276. doi: 10.1007/s11277-008-9521-2

[51]. UgoColesanti ,SapienzaUniversità Di Roma , &Santini, S. (2010). *A Performance Evaluation Of The Collection Tree Protocol Based On Its Implementation For The Castalia Wireless Sensor Networks Simulator*. Paper presented at the Proceedings of the 6th ACM conference on embeddednetwork sensor systems, Raleigh, NC, USA.

[52]. UCI. (2011). A Comparative Review of WSN Retrieved 26/09/2011, from https://eee.uci.edu/wiki/index.php/A_Comparative_Review_of_WSN

[53]. Singh, C. P., Vyas, O. P., &Tiwari, M. K. (2008). *A Survey of Simulation in Sensor Networks*. Paper presented at the Proceedings of the 2008

International Conference on Computational Intelligence for Modelling Control \& Automation.

[54]. Kpke, A., Swigulski, M., Wessel, K., Willkomm, D., Haneveld, P. T. K., Parker, T. E. V. Valentin, S. (2008). *Simulating wireless and mobile networks in OMNeT++ the MiXiM vision*. Paper presented at the Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems \& workshops, Marseille, France.

[55]. Wessel, K., Swigulski, M., Kpke, A., &Willkomm, D. (2009). *MiXiM: the physical layer an architecture overview*. Paper presented at the Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Rome, Italy.

[56]. Jun Zheng and Abbas Jamalipour. (2009). *Wireless Sensor Networks: A Networking Perspective*. Wiley-IEEE Press.

[57]. Olivares, T., Tirado, P. J., & Orozco-Barbosa, L. (2006). *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.

[58]. Trung, Q. L., & Kotsis, G. (2008). Arch-AdSenNets: an architecture for inter-working MANET with wireless sensor and actor networks. *Int. J. Commun. Netw. Distrib. Syst., 1*(4/5/6), 466-506. doi: 10.1504/ijcnds.2008.021080

[59]. Li, X. J., Seet, B.-C., & Chong, P. H. J. (2008). Multihop cellular networks: Technology and economics. *Comput. Netw., 52*(9), 1825-1837. doi: 10.1016/j.comnet.2008.01.019

[60]. Whitaker, R. M., Hodge, L., & Chlamtac, I. (2005). Bluetooth scatternet formation: A survey. *Ad Hoc Netw., 3*(4), 403-450. doi: 10.1016/j.adhoc.2004.02.002

[61]. Kim, B.-K., Hong, S.-K., Jeong, Y.-S., & Eom, D.-S. (2008). *The Study of Applying Sensor Networks to a Smart Home*. Paper presented at the Proceedings of the 2008 Fourth International Conference on Networked

Computing and Advanced Information Management - Volume 01.

[62]. Mpitziopoulos, A., Konstantopoulos, C., Gavalas, D., & Pantziou, G. (2008). *Hazard monitoring for visually impaired people enabled by wireless sensor networking technology*. Paper presented at the Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments, Athens, Greece.

[63]. Rhee, I., Warrier, A., Aia, M., Min, J., & Sichitiu, M. L. (2008). Z-MAC: a hybrid MAC for wireless sensor networks. *IEEE/ACM Trans. Netw., 16*(3), 511-524. doi: 10.1109/tnet.2007.900704

[64]. Sharif, A., Potdar, V., & Rathnayaka, A. J. D. (2009). *Performance evaluation of different transport layer protocols on the IEEE 802.11 and IEEE 802.15.4 MAC/PHY layers for WSN*. Paper presented at the Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia, Kuala Lumpur, Malaysia.

[65]. Casey, P. R., Tepe, K. E., & Kar, N. (2010). Design and Implementation of a Testbed for IEEE 802.15.4 (Zigbee) Performance Measurements. [Article]. *EURASIP Journal on Wireless Communications & Networking*, 1-11. doi: 10.1155/2010/103406

[66]. Ramachandran, I., Das, A. K., & Roy, S. (2007). Analysis of the contention access period of IEEE 802.15.4 MAC. *ACM Trans. Sen. Netw., 3*(1), 4. doi: 10.1145/1210669.1210673

[67]. Gay, D., Levis, P., & Culler, D. (2007). Software design patterns for TinyOS. *ACM Trans. Embed. Comput. Syst., 6*(4), 22. doi: 10.1145/1274858.1274860

[68]. Ben-Othman, J., Diagne, S., Mokdad, L., & Yahya, B. (2010). *Performance evaluation of a hybrid MAC protocol for wireless sensor networks*. Paper presented at the Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems, Bodrum, Turkey.

[69]. Korkalainen, M., Sallinen, M., Krkk, N., inen, & Tukeva, P. (2009). *Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications*.

Paper presented at the Proceedings of the 2009 Fifth International Conference on Networking and Services.

[70]. Khan, M. Z., Askwith, B., Bouhafs, F., & Asim, M. (2011). *Limitations of Simulation Tools for Large-Scale Wireless Sensor Networks*. Paper presented at the Proceedings of the 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications.

[71]. Sommer, C., Dietrich, I., & Dressler, F. (2010). Simulation of Ad Hoc Routing Protocols using OMNeT++. *Mob. Netw. Appl., 15*(6), 786-801. doi: 10.1007/s11036-009-0174-5

[72]. Ugo Colesanti , Sapienza Università Di Roma , & Santini, S. (2010). *A Performance Evaluation Of The Collection Tree Protocol Based On Its Implementation For The Castalia Wireless Sensor Networks Simulator*. Paper presented at the Proceedings of the 6th ACM conference on Embedded network sensor systems, Raleigh, NC, USA.

[73]. Crossbow. (2011). IRIS module datasheet Retrieved 12/06/2011, from http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=264

[74]. Crossbow. (2011). MICA2 module datasheet Retrieved 12/06/2011, from http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=174

[75]. Crossbow. (2011). MICAZ module datasheet Retrieved 12/06/2011, from http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=164

[76]. Crossbow. (2011). TelsoB module datasheet Retrieved 12/06/2011, from http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=252

[77]. Crossbow. (2011). Imote2 module datasheet Retrieved 12/06/2011, from http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=253