Power Characterisation of a Zigbee Wireless Network in a Real Time Monitoring Application

Arrian Prince-Pike

A thesis submitted to Auckland University of Technology in fulfilment of the requirements for the degree of Master of Engineering (ME)

April 2009

School of Engineering

Primary Supervisor: Dr John Collins

Acknowledgements

In completing this research I received help from a number of people. First I must thank my supervisor Dr John Collins for his support, guidance and advice during the course of this investigation.

I must also thank AUT technician Achala Perera who provided me with much of the equipment required for the experimental investigation.

Finally, thanks to my friends and family for their support during the course of my research.

Attestation of Authorship

'I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning'

Arrian Prince-Pike
April 2009

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
AODV	Ad-hoc On-demand Distance Vector
ARP	Address Resolution Protocol
BI	Beacon Interval
BO	Beacon Order
CAP	Contention Access Period
CFP	Contention Free Period
CPU	Central Processing Unit
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
FFD	Full Function Device
GTS	Guaranteed Time Slots
IEEE	Institute of Electrical and Electronic Engineers
kbps	kilobits Per Second
LCD	Liquid Crystal Display
MAC	Media Access Control
NS-2	Network Simulator version 2
OSI	Open Standards Interconnection
OTCL	Object TCL
PAN	Personal Area Network
PAN ID	Personal Area Network Identifier
PC	Personal Computer
PHY	Physical (Refers to the Physical network layer)
RAM	Random Access Memory
RF	Radio Frequency
RFD	Reduced Function Device
RX	Receive
SD	Superframe Duration
SO	Superframe Order
TCL	Tool Command Language
TX	Transmit
UART	Universal Asynchronous Receiver Transmitter
ZC	Zigbee Coordinator
ZED	Zigbee End Device
ZR	Zigbee Router

Abstract

Zigbee is a relatively new wireless mesh networking standard with emphasis on low cost and energy conservation. It is intended to be used in wireless monitoring and control applications such as sensors and remotely operated switches where the end devices are battery powered. Because it is a recent technology there is not sufficient understanding on how network architecture and configuration affect power consumption of the battery powered devices.

This research investigates the power consumption and delivery ratio of Zigbee wireless mesh and star networks for a single sink real time monitoring system at varying traffic rates and the beacon and non beacon mode operation of its underlying standard IEEE 802.15.4 in the star network architecture.

To evaluate the performance of Zigbee, the network operation was simulated using the simulation tool NS-2. NS-2 is capable of simulating the entire network operation including traffic generation and energy consumption of each node in the network. After first running the simulation it was obvious that there were problems in the configuration of the simulator as well as some unexpected behaviour. After performing several modifications to the simulator the results improved significantly.

To validate the operation of the simulator and to give insight on the operation of Zigbee, a real Zigbee wireless network was constructed and the same experiments that were conducted on the simulator were repeated on the Zigbee network. The research showed that the modified simulator produced good results that were close to the experimental results.

It was found that the non beacon mode of operation had the lowest power consumption and best delivery ratio at all tested traffic rates.

The operation of Zigbee mesh and star networks were compared to the results for IEEE 802.15.4 star networks in non beacon mode which revealed that the extra routing traffic sent by the Zigbee networking layers does contribute significantly to the power consumption, however even with the extra routing traffic, power consumption is still so low that it the

battery life of the device would be limited by the shelf life of the battery, not by the energy consumption of the device.

This research has successfully achieved its objectives and identified areas for future development. The simulator model for NS-2 could be improved to further increase the accuracy of the results as well as include the Zigbee routing layers and the experimental results could be improved by a more accurate power consumption data acquisition method.

Chapter 1 - Introduction

Zigbee is a recent wireless mesh networking standard with emphasis on low cost and energy conservation. It is intended to be used in wireless monitoring and control applications such as sensors and remotely operated switches where the end devices are battery powered. Currently, it is able to operate at the 868 MHz band at a data rate of 20 kbps in Europe, 916 MHz in the USA at 40 kbps and 2.4 GHz at 250 kbps world wide. Its primary features are:

- Standards based technology
- Worldwide usability at 2.4 GHz, no need for locale specific products
- Interoperability with other vendors. (This is one of the key reasons wireless computer networking has become so popular)
- Very low power consumption
- Small software stack footprint
- Support for large networks, up to 65 thousand devices
- Built in security and encryption
- Reliable mesh networking

The Zigbee networking protocol was developed in response to the unsuitability of current wireless protocols such as WIFI and Bluetooth for applications which require very long battery life, high reliability and low data rates such as remote monitoring and building automation. The first Zigbee specification was ratified in late 2004 and is based upon the physical (PHY) and media access control (MAC) layers of IEEE standard 802.15.4. Typical applications for Zigbee mesh networks include:

Home Control: Smart lighting, Entertainment, Climate control
Commercial buildings: Energy monitoring, Heating and ventilation, Lighting, Access control
Industrial: Process control, Environment management

1.1 Literature Review

Since Zigbee and its underlying standard IEEE 802.15.4 are recent, there has been little research investigating the power consumption of the different network architectures and comparison between the possible operating modes of the two networks. Most studies focus on beacon enabled network mode because most applications require bi-directional data flow and in star networks it gives the best energy savings. However in a real time monitoring application, data flow is mostly one way and is sent from the sensor node to a central storage device for processing and recording and therefore beacon enabled mode might not give the best energy savings.

Zheng and Lee [1] developed a computer simulation model of the media access control (MAC) and physical (PHY) layers of IEEE 802.15.4 for the Network Simulator-2 (NS-2)[2] to quantify its operation. They then studied its operation in star and peer-to-peer network topologies in beacon mode with focus on association efficiency, orphaning, collisions and duty cycle. Beacon mode is a mode of operation of IEEE 802.15.4 where network nodes are synchronised by periodic packet broadcasts or 'beacons' which are sent out by the coordinator. Their research shows that IEEE 802.15.4 is an excellent low power low data rate wireless standard upon which applications can be built. The simulation model created by Zheng and Lee has been the subject of a large number of research projects, several of which are detailed in the following paragraphs.

Several studies investigated the performance and power consumption of Zigbee and the IEEE 802.15.4 standard. Huang and Pang [3] recently investigated the effect of two system parameters in the MAC layer of a Zigbee network, beacon order and superframe order, on the power consumption of a small star architecture beacon enabled Zigbee network. The beacon order and superframe order are two parameters which set the duty cycle of the network nodes. Their work showed that these parameters can considerably increase the power consumption if the parameters are set incorrectly.

A beacon enabled star network was also researched by Ling-xi *et al*. [4]. They created a state diagram detailing the different states the transceiver can be in along with time intervals and current measurements for a Freescale MC13192 transceiver. Their research shows that

adjusting the transmission power does not have a large effect on overall power consumption because the duration spent in transmit mode is small. It also illustrates that increasing the number of nodes in the network results in higher average power consumption due to the extra network contention.

Similarly, research by [5] shows through an analytical model and NS-2 simulations that increasing the number of nodes in the network results in higher power consumption.

Research by Kohvakka *et al* [6] investigated the performance and power consumption of a large beacon enabled cluster tree (a collection of small star networks) Zigbee network. Their analysis extensively modelled the power consumption of a Chipcon CC2420 transceiver and determined that minimum power is consumed by end devices and co-ordinators when the beacon interval was set to 3.93 seconds and data was transmitted every 4 minutes. Their research, like [2], showed that the correct setting of the beacon order and superframe order has a significant effect on network performance and power consumption.

The study in [7] focuses on configuring the IEEE 802.15.4 MAC layer for the star network architecture. They modelled the performance of the beacon enabled and non beacon enabled modes of the MAC layer in an ideal star and non-ideal star networks. While the authors did not conclude whether beacon enabled mode uses less power than non beacon mode, it can be seen from their simulations that there is a higher percentage of packets received when using beacon enabled mode which would mean less retransmissions and subsequently lower power consumption.

Extensive research, experiments and modelling were performed in [8] to optimise power consumption in IEEE 802.15.4 networks. As in [9], they performed extensive characterisation of a Chipcon CC2420 radio and used this data in network simulations to determine power consumption. As part of their research, they performed a power breakdown of energy consumed by different parts of the protocol at the receiver. Interestingly this shows that less than 50% of the power is consumed by the actual transmission, most is consumed in the contention process and waiting for a beacon.
Research performed by [10] was one of the few studies that focused on real Zigbee devices. They investigate beacon and non beacon mode on Chipcon CC2420 and Freescale MC13192 development boards. Their research focused on investigating data rate and delivery ratio

under the influence of different beacon orders, number of nodes and varying data packet sizes. Although their research only focused on the situation of beacon order equalling superframe order (duty cycle 100%) their research shows that a higher throughput is possible in non beacon mode. Their research also showed that at lower data rates, increasing the number of nodes in non-beacon networks only results in a small drop in delivery ratio.

Burchfield *et al*. [11] investigated maximum throughput in a Zigbee network. In order to determine the maximum throughput they approached it in three parts; theoretical analysis, NS-2 simulations and hardware testing on Ember EM2420 development boards. Their work showed that the theoretical analysis achieved a maximum throughput that was almost the same as NS-2, however the hardware testing results initially were very different, achieving less than half the throughput of the simulation or theoretical analysis. After performing extensive modifications to the firmware on the development boards they achieved a maximum throughput that was 10% less than the simulation or theoretical analysis. This suggests that practical results in this research could differ considerably from the simulations without extensive modifications to either the firmware on the development boards or even the simulator itself.

Singh *et al*.[12] analysed the performance of the star network topology for monitoring and control applications. They created a model for the 802.15.4 MAC using a Markov renewal process which was then validated using NS-2. Their work highlighted several inconsistencies between the NS-2 IEEE 802.15.4 simulator model and the IEEE 802.15.4 standard. They compared their analytical model to the NS-2 simulator while varying a large number of parameters such as traffic rate, number of nodes and packet discard probability. They also analysed the power consumption of a node. Their research interestingly shows that the clear channel assessment functionality in the 802.15.4 MAC consumes a significant amount of energy, particularly at higher data rates.

Rao [13] performed extensive analysis on a beacon enabled IEEE 802.15.4 star network using NS-2. His research highlighted several issues with the NS-2 simulation model which were resolved. From his research he developed an adaptive back-off mechanism for the carrier sense multiple access with collision avoidance algorithm (CSMA/CA) used in IEEE 802.15.4 which improved throughput and delivery ratio considerably. Battery life was also investigated which showed that IEEE 802.15.4 nodes are likely to have a long battery life.

A beacon enabled IEEE 802.15.4 cluster tree network for home area networking was investigated by [14]. They calculated the power consumption of a cluster tree network using the measured power consumption data of a Chipcon CC2430 transceiver. They then calculate the expected battery life of a network node based on a 50 mAH battery. Their results show that the battery life would be around 240 days at a 1 minute transmission interval at a beacon order of 6 and superframe order 0. They state that a beacon enabled cluster tree network is optimum for home area networks without providing comparison to non beacon networks or other network topologies.

The authors of [15] focused on a temperature monitoring system for a power station using Zigbee. They used custom made nodes utilising a Chipcon CC2430 transceiver and Zigbee mesh networking with one coordinator, two routers and eight end devices. They investigated the power consumption of each end device and showed that there is very little difference in power consumption between each node. This is useful because it means that power consumption in a Zigbee network could be accurately determined by just measuring one node which would save a significant amount of time.

Currently there is insufficient data comparing beacon and non beacon network configurations as well as the difference between the star and mesh network architectures. This knowledge is important when setting up a Zigbee network to prevent excess power consumption and reduced battery life especially in a real time monitoring application.

1.2 Objectives and Methodology

This research is part of ongoing work to implement a Zigbee low power wireless real time monitoring system for use in cool stores and other industries where the key aims are high data reliability and a battery life of at least one year.

Zigbee is a highly configurable wireless low power low data rate standard. It supports multiple network topologies and many parameters which need to be configured depending on the application in which it is used.

From the literature review, it is important to investigate how network architecture and configuration affects power consumption in order for it to be minimised. For the proposed research, the power consumption, battery life and delivery ratio of end devices in the star and mesh network architectures will be modelled by simulating the operation of the Zigbee network architecture on a computer using simulation and modelling software. The simulations will involve simulating different traffic rates on the network in star and mesh topology as well as in beacon and non beacon mode.

The power consumption and subsequent battery life can be estimated by using the current consumption figures for the operating modes of a real transceiver. Following the simulations a small Zigbee network using hardware evaluation boards will be constructed with 5 to 10 devices for evaluating the results of the simulation. Extensive analysis of the results will be undertaken which will provide guidance as to how to best configure a Zigbee network for minimal power consumption while maintaining a good delivery ratio. Further work will involve developing an application which dynamically adjusts the transmission power of the Zigbee transmitter to further decrease power consumption.

After the analysis of data the results will be implemented on a production grade platform and network to further verify the validity of the results

The aims of this research are to:

- Determine the optimum Zigbee network configuration which minimises power consumption while obtaining good delivery ratio which results in a battery life of at least one year.

- Validate results from computer simulations with a real life Zigbee network to provide analysis on the accuracy of the simulator model.

Chapter 2 – Zigbee and IEEE 802.15.4 Overview

2.1 Introduction

This chapter gives an overview of Zigbee wireless networks and the underlying IEEE standard 802.15.4. It covers the key components that make up IEEE 802.15.4 and explains the relationship between Zigbee and 802.15.4. The different network topologies that are possible with IEEE 802.15.4 are illustrated as well as the different operational modes available. Key network parameters used in this research are also discussed.

2.2 IEEE 802.15.4

IEEE 802.15.4 was developed due to the unsuitability of current wireless standards such as WI-FI (802.11) and Bluetooth (802.15.1) for low data rate battery powered ad hoc networks. The IEEE standard 802.15.4 for low data rate personal area networks was finalised in May 2003 with a revised standard being issued in 2006.

The standard specifies the MAC and PHY layers of the open standards interconnection (OSI) network model while leaving the development of the upper layers to the designer. Other protocols such as Zigbee, WirelessHART and 6LowPAN offer a complete network solution by developing the upper layers of the model to include additional services not covered by the standard such as mesh networking, routing and security encryption.

2.2.1 Device types

The standard defines two types of network nodes

Full function devices (FFD): Full function devices can be configured to be the coordinator for the network, in which case it is typically called a personal area network (PAN) coordinator. FFDs are capable of communicating with any other device.

Reduced function devices (RFD): Reduced function devices are intended to be extremely simple devices with minimal hardware and software resources. RFDs can only communicate with FFDs or the PAN coordinator, not with another RFD.

2.2.2 Network Topologies

IEEE 802.15.4 networks can be divided into two main topologies, star and peer-to-peer. A third topology is possible, called cluster tree which is a variation of the peer-to-peer topology.

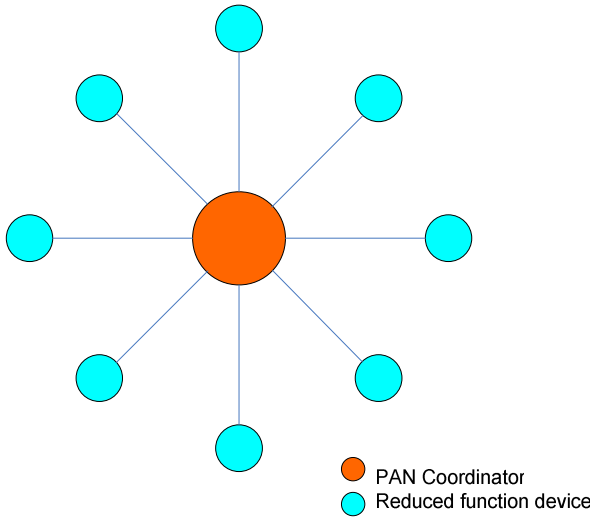


Figure 1 IEEE 802.15.4 star network topology

The star network topology is more structured than peer-to-peer. The PAN coordinator of the network is always the central node. Communication between devices always occurs via the PAN coordinator which relays messages between devices; direct messaging between devices is not permitted.

In the peer-to-peer topology, an arbitrary array of connections can be created between full function devices and the PAN coordinator. Since a network layer is not defined in the standard, routing is not directly supported.

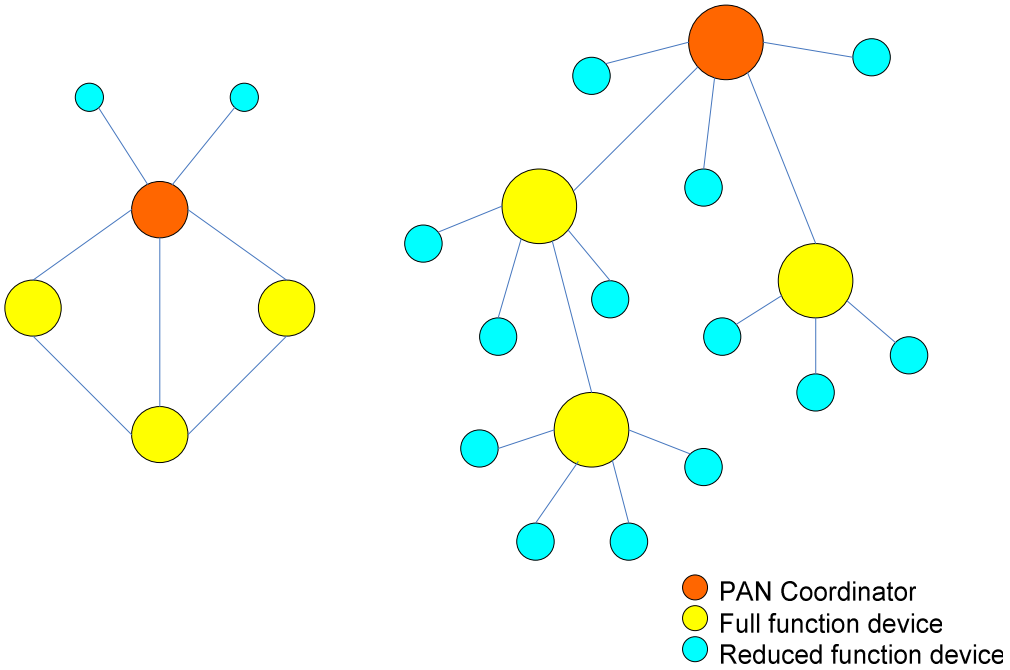


Figure 2 IEEE 802.15.4 peer-to-peer topology and IEEE 802.15.4 cluster tree topology

A cluster tree topology is also possible; this is a special case of the peer-to-peer topology. This exploits the fact that RFDs can only associate with FFDs and where most devices in the network are FFDs. Figure 2 illustrates the peer-to-peer and cluster tree topologies that are possible with IEEE 802.15.4.

2.2.3 Physical Layer

The physical layer is essentially the radio which converts the binary data to and from the MAC layer to RF energy to be radiated and received by the antenna. The physical layer also provides several key features which are utilised by the MAC layer, these include: channel energy detection, link quality index, channel selection and clear channel assessment.

The Physical layer in IEEE 802.15.4 is designed to operate in unlicensed frequency bands world wide. Since not all the frequency bands are the same world wide, IEEE 802.15.4 provides three possible operating frequencies, 868 MHz, 915 MHz and 2.4 GHz. Each frequency except 868 MHz has several channels which can be selected by the user. Table 1 lists the available frequencies and channels that can be used by IEEE 802.15.4.

Table 1 Available channels at different frequency bands and locations

Frequency band	Available channels	Data rate (kbps)	Symbol rate (ksymbols/s)	Locale
2.4GHz	16	250	62.5	World Wide
915 MHz	10	40	40	USA
868 MHz	1	20	20	Europe

For this research, the 2.4 GHz physical layer has been chosen due to the world wide availability of this frequency band and the wider availability of wireless transceivers at this frequency.

2.2.4 Data transport

A Frame is the basic unit of data transport in IEEE 802.15.4. There are four basic types used [16].

Data frame: Used for transferring application data between nodes
Acknowledgement frame: Used to confirm successful frame reception and are optional
Beacon frame: Used by FFDs to transmit beacons
MAC command frame: As their name suggests, used to send commands to devices in the network such as Associate request, Disassociate notification, Data request and others. RFDs only support a limited number of MAC commands due to their very nature.

A superframe structure can optionally be used which provides synchronisation to network nodes through the use of beacons which are described in 2.5.5.1.

2.2.5 MAC Layer

The MAC layer is responsible for a number of tasks. These include addressing data, beacon management, guaranteed time slots (GTS) management, acknowledged frame delivery, network association and disassociation. For incoming data it determines the address of the source and for outgoing data it determines the destination address. The MAC layer also assembles incoming packets and controls access to the physical layer.

2.2.5.1 Superframe Structure

IEEE 802.15.4 networks are able to operate in two different modes of operation, beacon mode or non beacon mode. In beacon mode, the network is fully synchronised as the coordinator sends out periodic packets or beacons. The purpose of this is to enable all nodes to sleep between beacons and wake up when the beacon timer expires, ready to receive the beacon from the coordinator.

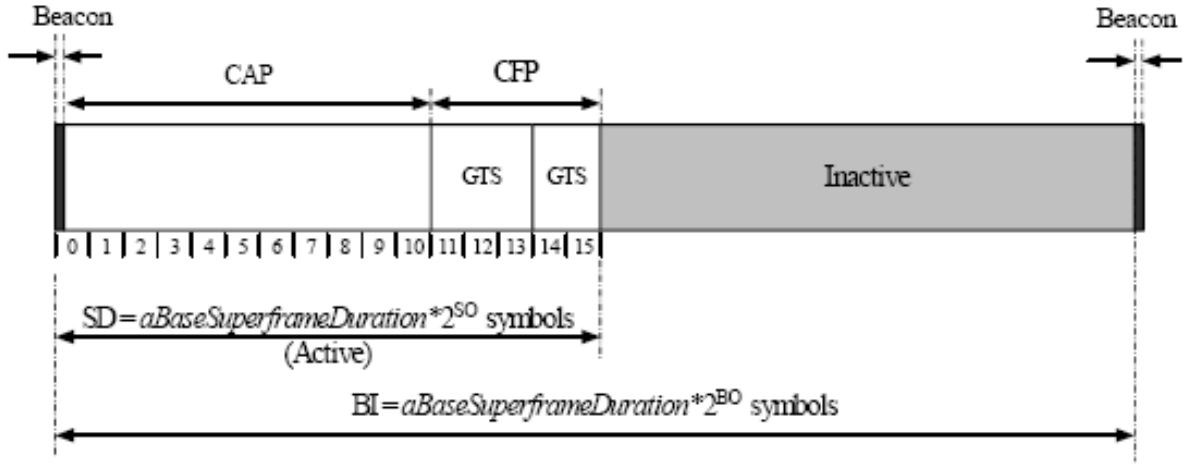


Figure 3 An example of superframe structure [16]

The superframe structure is only applicable in beacon mode networks. In non beacon networks the superframe structure is disabled and nodes contend for channel access by CSMA/CA. The duration of the superframe is divided into 16 slots with the beacon being transmitted in the first slot and the rest of the superframe available for nodes to communicate. The structure of the superframe is determined by the coordinator and consists of two sections, the contention access period (CAP) and an optional contention free period (CFP, also known as guaranteed time slots or GTS). The coordinator determines how many slots are allocated for CAP and CFP depending on how it has been configured. A sample superframe structure is illustrated in Figure 3.

The contention access period is the number of available slots where nodes use the slotted CSMA/CA protocol to contend for a slot to access the channel to transmit data. The contention free period or guaranteed time slots is the duration for which certain nodes which require low latency for sending data are reserved a slot which enables them to directly transmit data. There can be as many as 7 slots reserved for guaranteed time slots. The contention free period immediately follows the contention access period [17].

The inactive portion of the superframe structure is the period where devices in the network including the coordinator power down their transceiver and go to sleep. During this time there are no transmissions. This differs from non beacon networks where the coordinator and routers must remain on with only end devices being allowed to sleep.

The composition of the superframe structures duty cycle is determined by two key parameters, beacon order (BO) and superframe order (SO). The beacon order is used to calculate the beacon interval (BI) which is the duration between successive beacons and the superframe order is used to calculate the superframe duration (SD) which is the length of the superframe's active period. The beacon and superframe orders can range from 1 to 15 inclusive but the superframe order must be less than or equal to the beacon order. When the beacon and superframe order are equal there is no inactive period between beacons, when the superframe order is less than the beacon order there is an inactive period and setting them both to 15 disables beacon mode. The beacon and superframe order parameters therefore determine the duty cycle of the nodes in the network.

The beacon interval and superframe duration are related to beacon order (BO) and superframe order (SO) by the following equations:

$$BI = aBaseSuperframeDuration \times 2^{BO}$$
$$SD = aBaseSuperframeDuration \times 2^{SO} \qquad [17]$$

$$aBaseSuperframeDuration = aBaseSlotDuration \times aNumSuperframeSlots$$

According to the IEEE 802.15.4 standard[16],
$aBaseSlotDuration = 60$ symbols and $aNumSuperframeSlots = 16$, therefore $aBaseSuperframeDuration = 960$ symbols.

BI and SD can be calculated for a particular BO and SO as follows, for example using BO = 8 and SO 4:

$$BI = aBaseSuperframeDuration \times 2^{BO} \qquad SD = aBaseSuperframeDuration \times 2^{SO}$$
$$BI = 960 \times 2^{8} \qquad SD = 960 \times 2^{4}$$
$$BI = 245760 \text{ Symbols} \qquad SD = 15360 \text{ Symbols}$$

To convert BI from symbols to time in seconds, BI is divided by the symbol rate, using Table 1, at 2.4GHz it can be seen that the symbol rate is 62.5 ksymbols/second. Therefore,

$$BI = \frac{245760}{62.5 \times 10^{3}} \qquad SD = \frac{15360}{62.5 \times 10^{3}}$$
$$BI = 3.93216 \text{ Seconds} \qquad SD = 0.24576 \text{ Seconds}$$

Table 2 shows the calculated beacon interval and superframe durations for each frequency band.

Table 2 Beacon interval and superframe duration in seconds at different beacon orders and frequencies [7]

BO & SO	868 MHz	915 MHz	2.4 GHz
0	0.048	0.024	0.01536
1	0.096	0.048	0.03072
2	0.192	0.096	0.06144
3	0.384	0.192	0.12288
4	0.768	0.384	0.24576
5	1.536	0.768	0.49152
6	3.072	1.536	0.98304
7	6.144	3.072	1.96608
8	12.288	6.144	3.93216
9	24.576	12.288	7.86432
10	49.152	24.576	15.72864
11	98.304	49.152	31.45728
12	196.608	98.304	62.91456
13	393.216	196.608	125.82912
14	768.432	393.216	251.65824

2.2.6 Network Reliability

The physical layer utilises the CSMA/CA protocol to help avoid collisions with other nodes on the network. There are slotted and unslotted versions of CSMA/CA and the version used depends on whether beacons are enabled or not.

In non beacon mode, unslotted CSMA/CA is used which involves listening to the medium for activity and if activity is detected a random exponential back-off algorithm is used which waits for a random time before trying to access the channel again. If after a defined number of retries the channel is still busy, the PHY reports to the MAC layer that transmission failed.

Acknowledge command packets are another feature in IEEE 802.15.4 to improve network reliability. These are optional but if enabled they easily enable the sending node to determine if the destination node received the data packet correctly.

2.3 Zigbee

Zigbee is a wireless mesh networking standard which is based on the MAC and PHY layers of IEEE standard 802.15.4. Its main emphasis is on low cost, low power consumption, standards interoperability but with advanced features such as mesh networking and security.

The niche market for Zigbee involves applications requiring low data rates and low power consumption as well as advanced features such as mesh networking. Applications that would benefit from Zigbee include areas such as industrial control, building and home automation, environment monitoring and many others.

2.3.1 Software Architecture

The Zigbee software stack implements its functionality using a layered approach similar to the OSI layer model. The stack basically consists of three main layers:

- Application layer
- Zigbee stack/network layer
- Physical/Data link layer

The application layer contains applications that run on the node which give the device its functionality. Examples of applications could be one that measures temperature, another to measure humidity and another application to send the results to the coordinator.

The Zigbee stack layer sits above IEEE 802.15.4 and provides most of the network functionality for the network. This layer is responsible for network formation (if the device is a coordinator), assigning network addresses, message routing, security and implementing route discovery.

The physical/data link layer of Zigbee is based on IEEE 802.15.4 wireless network standard which itself consists of two layers, the MAC and PHY layers.

2.3.2 Node types

Zigbee networks can consist of three node types at the network level:

Zigbee Coordinator (ZC): Responsible for network formation and is the most capable device on the network. The coordinator forms the root of the network and is frequently an interface to other networks. There must be one (and only) coordinator per Zigbee network, regardless of network topology. It also stores security keys for the network if they are being used.

Zigbee Router (ZR): Acts as an intermediate between two devices which relays information between nodes. Routers also assign addresses to devices which are its children.

Zigbee End Device (ZED): Is the simplest Zigbee node type. It contains enough functionality to communicate with its parent node (either a coordinator or router) as well as perform its intended function and cannot relay data to other nodes. Because of this, the ZED is able to spend most of the time asleep conserving power, waking up to take measurements and transmit or receive data and then returning to sleep.

To take full advantage of the mesh networking capabilities of Zigbee, most Zigbee networks will contain all three node types.

2.3.3 Network architecture and topologies

Zigbee networks can be configured into three topologies:

Star

The star topology is the simplest and has the most limited functionality of the three Zigbee topologies. A star network consists of one coordinator and a number of end devices, see Figure 4. End devices are not able to communicate directly with each other and can only communicate via the coordinator. The Zigbee star network works the same as the IEEE 802.15.4 star network.

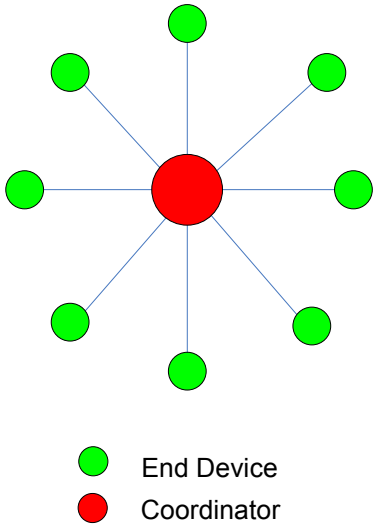


Figure 4 Zigbee star network topology

A problem with this topology is that there is no alternative route if the main link fails between the end device and the coordinator.

Cluster tree

A cluster tree topology consists of a coordinator which is linked with a number of child nodes which can either be end devices or routers. However unlike the star topology, each router can be linked with a number of its own child devices, and thus the network branches out like a tree as illustrated in Figure 5.

In the cluster tree topology, child devices are only able to communicate with their parent device. Parent devices can only communicate with their child nodes and their own parent device. When sending messages around the network, the message travels up the tree through the parent nodes until it reaches the common parent of the destination node, then travels back down the tree to the destination node.

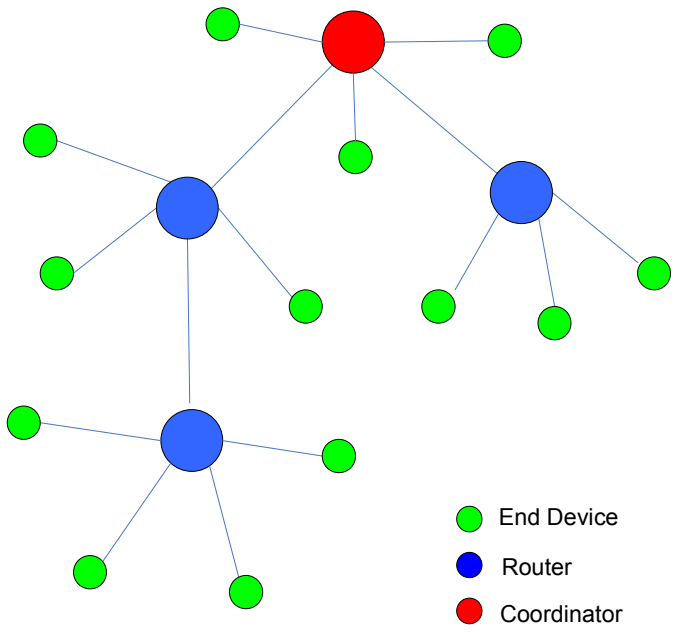


Figure 5 Zigbee cluster tree topology

A problem with this topology, like the star, is that there is no alternative route if the main link fails or becomes congested.

Mesh

The mesh network topology is the most useful Zigbee network topology. While it looks similar to the cluster tree topology, routers are able to communicate directly with each other if they are in range, thus forming multiple routes within the network. Figure 6 best illustrates the operation of the mesh network topology.

Because routers are able to communicate directly with each other, the mesh network topology has more efficient message propagation than a cluster tree and if one route in the network

fails, alternative routes can be found which increase reliability in the network and reduce congestion.

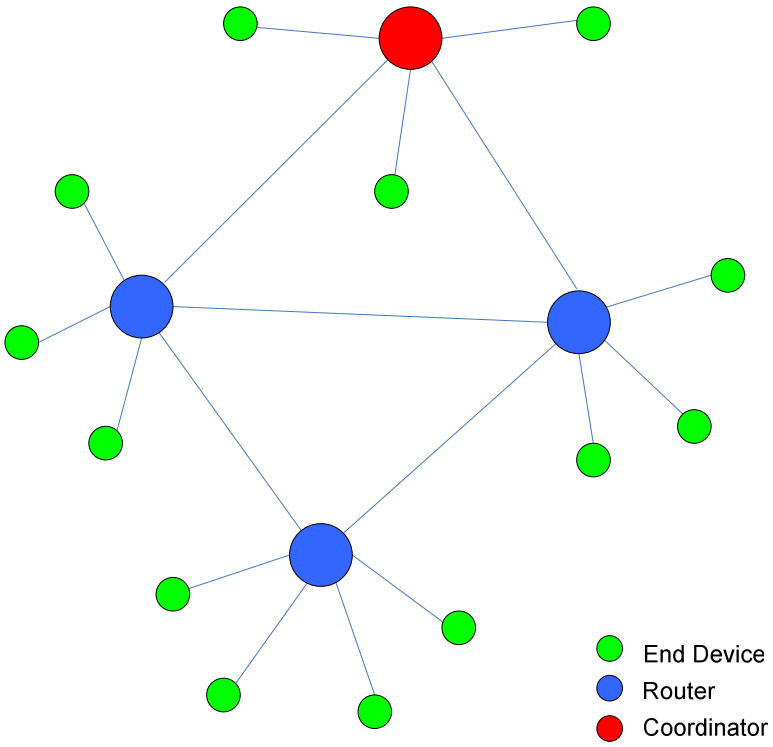


Figure 6 Zigbee mesh topology

The mesh network topology cannot be used when the Zigbee network is running in beacon mode, only star and cluster tree are possible in beacon mode because beacons cannot propagate through the network.

2.3.4 Network Addressing

Devices in a Zigbee network have two unique addresses which enable identification, an IEEE address and a network address.

The IEEE address is a unique 64 bit address assigned by the manufacturer which is allocated blocks of addresses by IEEE. This address is totally unique (like the MAC address in a PC network card) no two Zigbee devices in the world can have the same IEEE address. The IEEE address is also referred to as the MAC address or extended address.

The network address (also called short address) is a 16 bit address that identifies the node in a network. It is not unique therefore two devices in two different networks may be assigned the same network address.

Network addresses are assigned by the parent node (either the coordinator or router) when a node joins the network. One of the main uses of the network address is to reduce transmission overheads by only sending two byte addresses rather than eight bytes.

The coordinator in the network always has the network address of 0x0000.

2.3.5 Network Properties

There are a number of configurable parameters in a Zigbee network that must be configured on the coordinator before it can create a network. These parameters determine the size, topology and traffic flow of the network and are described in the following paragraphs.

2.3.5.1 Network Depth

Network depth is the maximum number of hops from the root of the network (the coordinator) to the most distant device in the network. This parameter therefore determines the size of the network. For star network topologies the network depth is one. A typical value for network depth is 5.

2.3.5.2 Number of Children

Each coordinator or router can have a number of child devices associated with it. This parameter defines the maximum number of child devices per parent. A common value in most Zigbee stacks is 20. This parameter is limited by the available RAM in the coordinator or router.

2.3.5.3 Number of Child Routers

This parameter puts a limit on how many routers a parent may have as children. This helps to prevent network bottle necks and limit network size.

The network depth, number of children and number of child router parameters are used by the Zigbee stack in the coordinator during initialisation to allocate blocks of network addresses to routers in the network. Routers then allocate addresses from these blocks to their children.

2.3.5.4 Beacons

Zigbee networks can operate in either beacon or non beacon mode. In beacon mode, the network is fully synchronised as the coordinator sends out periodic packets or beacons. The operation of beacons has been described in Section 2.2.5.1.

When nodes are searching for a network to join, they listen for the beacons being emitted by the coordinator. If they are not found then the node broadcasts a join request. When nodes join a beacon network, they are configured to the same beacon order as the coordinator

For end devices to sleep between beacons, a MAC timer is programmed with the required delay calculated from the beacon order. After a beacon is received, the timer is loaded with the beacon interval and sleep mode is entered. The end devices wake up just before the coordinator is due to send a beacon to take into account clock drift of the end devices. Very long beacon intervals require accurate crystals to keep in synchronisation. Otherwise they will miss the beacon from the coordinator.

2.3.6 Network Formation

The coordinator is responsible for creating the Zigbee network, this consists of several steps.

Search Radio Channels: The coordinator can either search for a radio channel which has the least activity present or can start listening on a fixed channel.

Assign PANID: The coordinator then starts the network and assigns a unique identifier called a personal area network identifier (PAN ID) to the network. The PAN ID can either be pre-programmed or dynamically assigned by listening to other networks present on the same channel and randomly choosing a PAN ID that does not conflict with existing networks.

Final Configuration: The coordinator then finishes configuring itself, assigns itself the network address of 0x0000 and is then ready to respond to queries from other devices.

Chapter 3 - The Simulation Environment

3.1 Introduction

This chapter describes the work undertaken to simulate the operation of a Zigbee wireless network using computer network modelling software. Descriptive procedures of methods used in this research are covered, starting with Zigbee hardware research and simulator comparison through to preliminary results. After the simulator was set up, several sets of experiments were performed which highlighted problems with the simulator which are discussed in detail at the end of the chapter. The modifications described in this chapter were used to generate the final results in Chapter 5.

3.2 Zigbee Hardware Research

Since this research involves comparison between computer simulations and hardware experiments a suitable hardware platform needs to be chosen to be used in the experiments. Since this research has commercial implications there are several criteria that need to be met for a particular device to be chosen. Table 3 compares the Zigbee transceivers which are discussed below. A transceiver needs to be chosen before the simulations can begin as the simulator needs to be configured with the correct parameters to accurately model the Zigbee network operation.

Atmel

Atmel currently produce Zigbee transceivers and have recently started producing system on chip devices which combine one of their popular AVR 8 bit microcontrollers and an AT86RF230 transceiver in a single package. Atmel's transceiver has excellent RF performance and low power consumption and prior experience with Atmel microcontrollers would reduce development time. The main disadvantage with Atmel is the limited Zigbee software stack which is provided with the devices. The stack is missing some functionality.

Meshnetics

Meshnetics produce several Zigbee modules which are based on Atmel hardware. They appear to provide a more fully featured Zigbee stack than Atmel which includes a real time operating system to make writing applications easier as it helps to separate the Zigbee and user tasks. A problem with the supplied software is that most of the source code is hidden in pre-compiled libraries. Under normal circumstances this should not be a problem; however it could create problems if the functionality of the Zigbee stack needs to be modified for research purposes.

Chipcon – Now Texas Instruments

Chipcon is currently the most popular supplier of Zigbee transceivers and system on chip devices. Their devices have reasonable RF and power consumption performance and they provide a very full featured Zigbee and 802.15.4 MAC software with almost all the source code. This could be very useful if low level changes are required during the course of the research.

Microchip

Microchip currently only produces Zigbee transceivers. These devices have acceptable RF performance and power consumption. They provide a basic Zigbee stack for use with Microchip microcontrollers with full source code provided. Familiarity with Microchip microcontrollers makes this a possible choice as it would help speed development time however the limited functionality of their software stack is a major disadvantage.

Table 3 Zigbee transceiver comparison

Manufacturer	Product	ROM (KB)	Ram	Transmit power	Receiver sensitivity	Tx current	Rx current	Sleep current
Atmel	AT86RF230	NA	NA	3dBm	-101dBm	16.5mA	15.5mA	20nA
	ATmega128 RZAV	128	8KB	3dBm	-101dBm	26.5mA	25.5mA	1.02uA
Meshnetics	MNZB-24-A2	128	8KB	3dBm	-101dBm	18mA	19mA	6uA
	MNZB-A24-UFL	128	8KB	20dBm	-104dBm	50mA	23mA	6uA
Chipcon	CC2430	128	8KB	0dBm	-95dB	26.9mA	26.7mA	0.5uA
	CC2420	NA	NA	0dBm	-95dB	17.4mA	18.8mA	20nA
Microchip	MRF24J40	NA	NA	0dBm	-95dBm	23mA	19mA	2uA

After comparing the above devices, a Chipcon CC2430 device was chosen for this research. This was due to the following reasons:

- In built microcontroller
- Low cost
- Complete full featured Zigbee stack and 802.15.4 MAC layer software with source code
- Reasonable RF performance

3.3 Simulator comparison

There are several tools currently in use for simulating Zigbee wireless networks. The most common computer simulation tools are OPNET, OMNeT++ and NS2.

OPNET [18]

- Non free
- Some models open source
- Mature simulator, common in industry

OPNET is an advanced network simulator which is commonly used within industry due to its advanced features and user support base. However few research papers have been published using the IEEE 802.15.4/Zigbee model in OPNET.

OMNeT++ [19]

- Free for research or non profit purposes
- Open source
- More widely used by researchers compared to OPNET

OMNet++ has an IEEE 802.15.4/Zigbee wireless simulation model which was developed by Chen *et al*. [20]. However this model had only been released a short time before this research had commenced and because of this the model had little use by other researchers.

NS-2 [2]

- Free
- Open source
- Widely used

NS-2 has been available for a long time and has been used by a large number of researchers world wide. The IEEE 802.15.4/Zigbee simulation model for NS-2 was developed by Zheng and Lee [21] in 2005 and has been the focus for many researchers with many published papers using this simulator [1, 7, 11, 13]. Currently the most commonly used and most mature computer simulation tool for Zigbee and IEEE 802.15.4 networks is NS-2.

For this research, NS-2 has been chosen for the network simulator because it is the most widely used, it is free and fully open source which is useful should modifications to the simulator be required.

3.4 Simulator overview

NS-2 is a discrete event simulator developed in a collaborative effort by many institutions and contains code contributions from many researchers. NS-2 can be built under any POSIX like operating system such as Linux or FreeBSD but can also be compiled under Microsoft Windows with Cygwin which provides a Linux like environment under Windows. For this research, NS-2 version 2.33 [2] has been used which was the most recent version at the time the research began.

The Zigbee 802.15.4 simulation model that has been used in this research and is now included in NS-2 was developed by Zheng and Lee [21] from the City College of New York. Ramachandran [22] also contributed many modifications to the original 802.15.4 simulation model which have been used in this research.

NS-2 uses two programming languages, C++ and objective tool command language (OTCL) in its development. C++ is used to write the simulation models and the core simulation environment due to its speed. OTCL is used as a front end scripting language and is used for writing simulation scripts as it is quicker to change for different simulation iterations and does not need to be recompiled.

NS-2 is typically controlled by a script file written in OTCL which contains commands for configuring the simulator. This script file creates a new simulation object and typically contains commands to perform the following tasks

- Create new simulator object
- Enable tracing
- Create network nodes and network topology
- Setup packet loss, link dynamics
- Create routing agent(s)
- Create traffic agent(s)
- Create null agent(s)
- Define finishing procedure
- Start simulation

The simulator object is the main class which supervises the entire simulation [23]. There is only one simulator object throughout the duration of the simulation. The simulator object also instantiates several objects when created, an important one being the scheduler object. The scheduler object is an event driven time management mechanism which simply executes the earliest scheduled event and on completion, executes the next event in the queue. An event consists of the time at which it is to be launched and an event handler function. One issue noted with the scheduler is the behaviour when two events are scheduled to be executed at the same time. Since the scheduler cannot execute two events simultaneously, it executes the events in the order that they were scheduled.

Nodes are abstract components which can act as hosts (a source or destination) or routers (an intermediate node) in the network. Nodes have interfaces to other network components and can have different network agents attached and detached such as routing agents, traffic generators and traffic sinks [23].

NS-2 is able to simulate the effects of different link dynamics such as link failures and packet loss. If this behaviour is desired it should be configured before the simulation starts.

Traffic and null agents are sources and sinks of network traffic respectively. Traffic agents output network packets at the rate, packet size and traffic type specified while a null agent acts as a sink for all packets, both successfully transmitted and dropped. Traffic and null agents are attached to nodes which make them behave as either a data source or data sink. There must be at least one data sink in the network.

The finish procedure is a set of instructions which are executed by NS-2 when the simulation ends. The finish procedure typically contains several instructions to flush any pending information to the trace file and close it. Depending on the user requirements the finish procedure can automatically start nam (the network animator) before NS-2 closes.

The last command in the trace file is a call to start the simulation which simply consists of: "$ns run"

3.4.1 Comments on NS-2

It was discovered during preliminary testing of NS-2 that that it did not fully support Zigbee, only the underlying IEEE 802.15.4 network layers. Therefore only star and cluster tree networks could be simulated. This was unfortunate as now only the star network topology could be compared to the experimental results, but it was decided to continue with NS-2 due to the large amount of prior research that had been undertaken by other researchers which suggests that it should perform well at modelling the star network topology.

3.5 Energy Model

An important consideration for wireless devices is power consumption. Because of this, NS-2 has an energy model to simulate the energy consumption of nodes. The energy model used in NS-2 is rather simple. When the simulation starts, each node is initialised with an initial starting energy in Joules using the `SetEnergy` function. The double variable `energy_` in the class holds the node's current energy level in Joules. The energy class definition [24] is shown below.

```
class EnergyModel : public TclObject {
public:
EnergyModel(double energy) {energy_ = energy;}
inline double energy() {return energy_;}
inline void setenergy(double e) {energy_ = e;}
virtual void DecrTxEnergy(double txtime, double P_tx)
{ energy_ -= (P_tx * txtime); }
virtual void DecrRcvEnergy(double rcvtime, double P_rcv)
{ energy_ -= (P_rcv * rcvtime); }
protected:
double energy_;
};
```

Whenever packet reception or transmission occurs, the energy_ variable is decremented using functions `DecTxEnergy` and `DecRcvEnergy`. These functions are passed two parameters which are the current consumption while transmitting or receiving respectively and the time spent transmitting or receiving. These functions then calculate the Joules of energy used and decrement the `energy_` variable by the calculated amount.

3.6 Zigbee simulation parameters

To get accurate results from the simulations, the simulator needs to be configured to match the hardware characteristics of the Chipcon transceiver. The key parameters that need to be configured are discussed below.

Transmit power consumption

Transmit power consumption is the power consumed during data transmission. The amount of power consumed is directly related to the size of the data packet as this determines how long the radio is transmitting. The transmit power consumption specified in the Chipcon CC2430 data sheet is listed as 26.9 mA. Since NS-2 uses watts in its energy model, the calculated power consumption using a 3V battery is:

$$P = VI$$
$$P = 3 \times 26.9 \times 10^{-3}$$
$$P = 0.0807W$$

Receive power consumption

Receive power consumption is the power consumed while the device is in receiving mode which for the CC2430 is 26.7mA. Using a 3V battery the receive power consumption is calculated to be:

$$P = VI$$
$$P = 3 \times 26.7 \times 10^{-3}$$
$$P = 0.0801W$$

Idle power consumption

Idle power consumption is the power consumed while the device is in idle/sleep mode. The idle power consumption is calculated to be:

$$P = VI$$
$$P = 3 \times 0.5 \times 10^{-6}$$
$$P = 1.5 \times 10^{-6} W$$

Antenna transmit and receive gain

The antenna that comes with the Chipcon CC2430 development board is a quarter wavelength monopole antenna. The gain specified by the manufacturer is 2.2 dBi [25], however the antenna propagation model in NS-2 uses natural numbers for gain measurements so the gain in dB is converted to a natural number.

$$G_0 = 10^{\frac{G_{dB}}{10}}$$
$$G_0 = 10^{\frac{2.2}{10}}$$
$$G_0 = 1.66$$

System loss

System loss is attenuation or loss of power not related to the propagation of the signal. System losses are typically due to transmission line attenuation, filtering losses and antenna losses. To simplify the practical testing and simulation comparison it is assumed that there is no signal attenuation ie system loss factor is 1.

Receiver sensitivity

Receiver sensitivity is the minimum input signal into the receiver which can be successfully decoded. The CC2430 has a receive sensitivity of -95 dBm. NS-2 expects the receiver sensitivity value in watts which requires the dBm value to be converted.

$$P_r = \frac{10^{\left(\frac{dBm}{10}\right)}}{1000}$$
$$P_r = \frac{10^{\left(\frac{-95}{10}\right)}}{1000}$$
$$P_r = 3.1623 \times 10^{-13} W$$

Radio propagation model

NS-2 supports three radio propagation models which are used to predict the received signal power of each packet and determine whether the signal power is above the receive threshold of the radio and can therefore be decoded successfully. The three models supported by NS-2 are the free space model, two ray ground model and the shadowing model.

Free space model

The free space model assumes ideal propagation conditions, there is a clear line of sight path between the transmitter and receiver and there is only one signal path between the transmitter and receiver. H.T Friis [24] produced the following equation for calculating received signal power at distance d from the transmitter.

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \text{ [24]}$$

Table 4 Terms in Friis equation

Variable	Meaning
Pr	Received power at distance d
Pt	Transmit power
Gt	Gain of transmitting antenna
Gr	Gain of receiving antenna
λ	Wavelength
d	Distance
L	System loss factor

Two ray ground model

Since a single line of sight signal path between nodes is unlikely to be the only means of propagation, the two ray ground model considers both a direct path and a ground reflection. It has been demonstrated that this model gives better results than the free space model at long distances [26]. The received signal power at distance *d* is given by

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L} \text{ [24]}$$

Where ht and hr are the heights of the transmitter and receiver antennas, the remaining parameters are the same as the free space model and are listed in Table 4.

Shadowing model

The free space and two ray ground models are mostly suitable for short range line of sight communications. However over large distances, the transmitted signal can undergo a number of effects such as fading and multi-path propagation which the first two models do not take into account.

The shadowing model consists of two parts, the first is known as the path loss model which predicts the mean received power $\overline{P_r(d)}$ at distance *d* relative to the power at an arbitrarily close-in distance d_o (e.g. 1 metre).

$$\left[\frac{\overline{P_r(d)}}{P_r(d_o)}\right]_{dB} = -10\beta \log\left(\frac{d}{d_o}\right) \text{ [24]}$$

Where beta is the path loss exponent, and is usually determined empirically by field measurements. Table 5 gives typical values of beta. $P_r(d_o)$ can be calculated by using the free space equation.

Table 5 Path loss exponent β at different environments [27]

Environment		**β**
Outdoor	Free space	2
	Shadowed urban area	2.7 to 5
Indoor	Line of sight	1.6 to 1.8
	Obstructed	4 to 6

The second part of the shadowing model reflects the variation of the received signal power at a certain distance and is a log normal random variable of Gaussian distribution designated by X_{db}. The shadowing model can therefore be represented by the following equation.

$$\left[\frac{\overline{P_r(d)}}{P_r(d_o)}\right]_{dB} = -10\beta \log\left(\frac{d}{d_o}\right) + X_{dB} \text{ [24]}$$

X_{db} is a Gaussian random variable with zero mean and standard deviation σ_{dB} which is called the shadowing deviation which is also determined empirically. Table 6 lists typical values of σ_{dB}.

Table 6 Typical values of shadowing deviation σ_{dB} [28]

Environment	**σ_{dB} (dB)**
Outdoor	4 to 12
Office, hard partition	7
Office, soft partition	9.6
Factory, line-of-sight	3 to 6
Factory, obstructed	6.8

These models are used to predict the received signal power of a packet. At the physical layer of the model is a receive threshold value. Each packet's power level must be above this threshold for the radio to be able to decode it successfully. For these simulations the two ray ground model has been chosen because it offers more accurate results than the free space model and since short distances are involved the shadowing model would be over complicating the scenario.

Therefore for the simulations in this investigation, the two ray ground model and the parameters listed in Table 7 have been used.

Table 7 Antenna model parameters used by NS-2

Parameter	**Symbol**	**Value**
Distance between nodes	d	10m
Gain of transmitting antenna	G_t	1.66
Gain of receiving antenna	G_r	1.66
Height of transmitting antenna	h_t	1.5m
Height of receiving antenna	h_r	1.5m
Transmit power	P_t	1mW
System loss	L	1

Initial energy

The energy model in NS-2 uses Joules as a measure of energy consumption, however most battery manufacturers specify energy capacity in ampere-hours. For the simulations and experiments in this thesis a CR2477 type Lithium Manganese Dioxide coin cell battery was

used which has a capacity of 900 mAH and a terminal voltage of 3 Volts which therefore gives an energy capacity of 9720 Joules.

Beacon and Superframe order

For this research a beacon order range of 6 to 10 has been chosen and a superframe order range of 0 to 2. A beacon order range of 6 to 10 results in a beacon interval of between 0.95 and 15.73 seconds which seemed appropriate based on the fastest transmission interval of 30 seconds/packet. In order to minimise power consumption the duty cycle of the network needs to be kept as low as possible, therefore the chosen superframe order range results in a CAP of between 15 and 62 ms long which seemed reasonable given the small network involved.

Data Rates

This research will investigate five different data rates which have been chosen to approximate likely reporting intervals for actual sensor end devices. The data rates (packet intervals) to be investigated are 30, 60, 100, 200 and 1000 seconds per packet.

3.7 Performance Metrics

Several metrics have been defined to characterise the performance of the simulations and hardware experiments. The metrics have been chosen to demonstrate the real world usability of the technology and how well it performs under different traffic loads and network configurations.

Delivery ratio

This is a percentage of data packets which are successfully received versus the number of data packets transmitted and is an important metric in determining how reliable the network link is. In these simulations the delivery ratio is calculated only for data packets and does not take dropped acknowledgement or other network service traffic into account.

Power consumption

This indicates the amount of power consumed by a node in milliwatts. This metric is the average power consumption by all nodes in the network excluding the coordinator.

Battery life

This is the average power consumption converted into a more human readable metric. The battery life is calculated by converting the amp-hour capacity of a chosen battery into Joules then dividing by the power consumed and converting the result from seconds into years. One of the key objectives of Zigbee is to have a long battery life measured in years. This metric will show whether the system will perform as intended and whether Zigbee is a suitable standard for low data rate wireless applications.

3.8 Simulation process

Initially several simulations were run by manually configuring the simulator from the command line followed by manual execution of a script file that analysed the trace file generated by NS-2 and created a text file which contained the result. It soon became apparent that this was a very time consuming way of generating results.

In order to use NS-2 efficiently, a number of scripts and programs have been written to automate the process of running the simulator with varying parameters for a large number of simulations, following with analysis on the resulting trace files to generate results. This section explains these scripts, programs and the process structure for generating the results presented in this thesis. The source code for the scripts and programs described in this section are available in Appendix D.

The diagram shown in Figure 7 shows the interdependencies between the simulator and processing scripts used to generate and process the resulting data in the simulation environment.

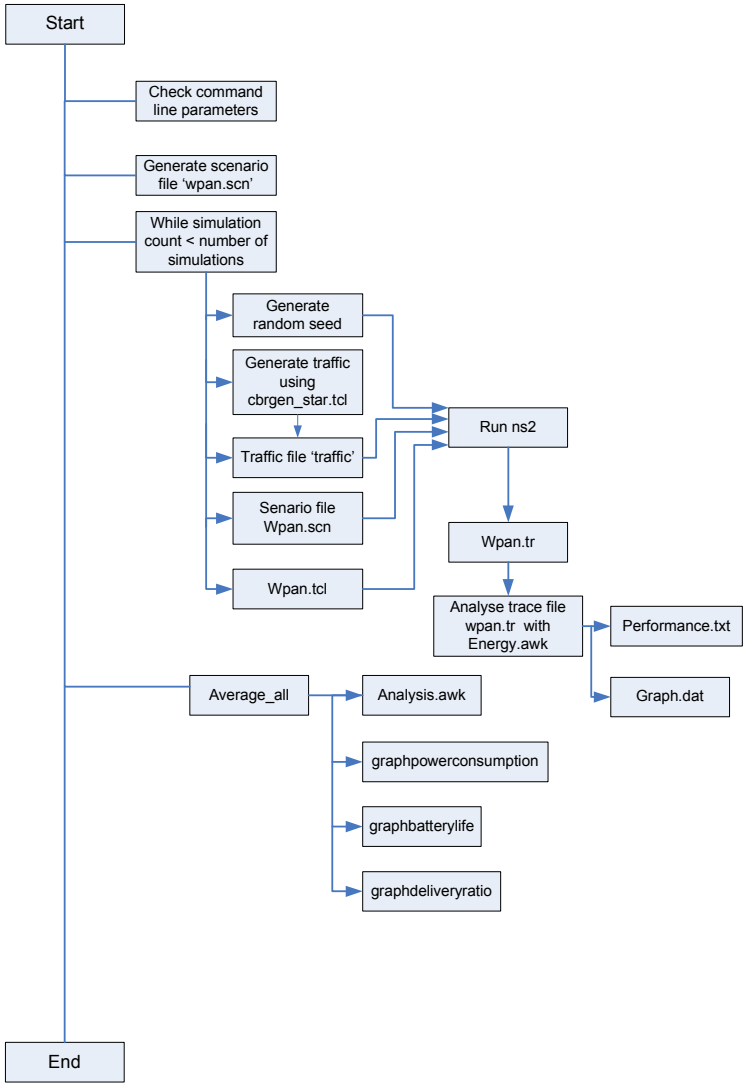


Figure 7 Process diagram for NS-2 simulations

`testpower`

The command line program `testpower` is a simple application written in C which automates the simulation process. It accepts the following command line parameters.

Table 8 Available command line parameters for the `testpower` program

Parameter	Purpose
-nobeacon	Disables beacon transmissions
-sims #sims	Number of simulations to perform
-maxbo BO	Maximum beacon order to simulate to
-nodes nn	Number of nodes in the network excluding coordinator
-startrate rate	Starting data rate (seconds/packet)
-stoprate rate	Final data rate (sec/packet)
-random	Enable randomised traffic delays
-logstep	Enable logarithmic traffic rate stepping

In order to produce conclusive results, the simulations should be generated in a random fashion taking the mean of a number of results. Since it is not practical to do this manually, this program automates this task.

When `testpower` is executed it reads the command line parameters passed to it and creates a new directory based on the current time and saves the command line parameters to a file in the directory. One of the parameters for the simulator is a random seed variable, this variable is generated by the `testpower` application each time a simulation is performed using random number functions provided by the computer. `Testpower` then calls a separate NS-2 traffic generator application to generate a traffic file which is used by `wpan.tcl` in the simulation.

Finally it executes the simulator with the simulation script, random seed, beacon order and superframe order as parameters.

Upon completion of the simulation, an analysis script is called which processes the trace file and generates statistics about the simulation which are saved in a file in the current directory.

This process is then repeated for each of the parameters specified at the command line.

`wpan.tcl`

This file is the main simulation script for NS-2. It contains all the simulation parameters discussed previously to configure the simulator such as transmit and receive power, number of nodes, receiver sensitivity and others. This file loads traffic and scenario files which are generated separately by other scripts for use with the simulation. Executing this script with NS-2 runs the simulation and produces a trace file which requires further processing.

`cbrgen_star.tcl`

`cbrgen_star` is a tool command language (TCL) script file which is used to generate traffic flows between nodes. This version has been modified from the original script by [13] and further modified for use in this research. The main modification has been to define the

communication direction. Because this research is focusing on how Zigbee performs in a real time monitoring application, traffic flows are only from the nodes to the coordinator. Therefore the coordinator is always the destination address for nodes in this traffic generator. Executing NS-2 with this script generates a text file which contains the traffic flows in the network which is used by `wpan.tcl`.

scen_gen

This program was developed by Rao [13] and generates the scenario file used by `wpan.tcl`. The file contains the locations in a three dimensional space of each node in the network and thus defines the physical layout of the network.

energy.awk

This file generates the performance metrics which are calculated by processing the trace file generated by NS-2 and saving the results to a file. This script was also developed by [13] but has been modified to calculate the performance metrics described in Section 3.7. This file is written in the awk scripting language which is efficient for processing text files which is the format of an NS-2 trace file. Two more scripts then average and plot the data.

`avg_througput` and `avg_power`

These two awk scripts average the performance metrics generated by `energy.awk` for each simulation. The files write the results to a comma separated value (CSV) file which can be opened using Microsoft Excel or OpenOffice for later viewing. These scripts also use `gnuplot` [29] to directly plot the results to a postscript file for immediate viewing or importing into another document.

3.9 Initial Simulation Results

The simulator was initially configured with the parameters listed in Table 9 and then executed with the `testpower` program. This produced the following results which are discussed in the following sections.

Table 9 Simulator configuration parameters

Parameter	Value
Beacon order	6-10 and 15(non beacon mode)
Superframe order	0-2 and 15(non beacon mode)
Time period between packets	30, 60, 100, 200 and 1000 seconds
Packet size	64 bytes
Simulation time	1000 seconds
Number of nodes	10 (excluding coordinator)
Transmit current consumption	26.9 mA
Receive current consumption	26.7 mA
Sleep power consumption	0.5 μA
Number of simulations	10
Distance between nodes and coordinator	10 metres
Random traffic jitter	Disabled
Energy capacity	9720 joules
Traffic direction	Node to coordinator
Network topology	Star

3.9.1 Beacon Mode

This section contains the battery life and delivery ratio results for beacon mode operation at data rates 30, 60, 100, 200 and 1000 seconds/packet which are illustrated in Figures 8 to 17. It was decided to exclude the power consumption graphs from this section since they are just the inverse of the battery life and are available in Appendix A.

Results at 30 seconds per packet

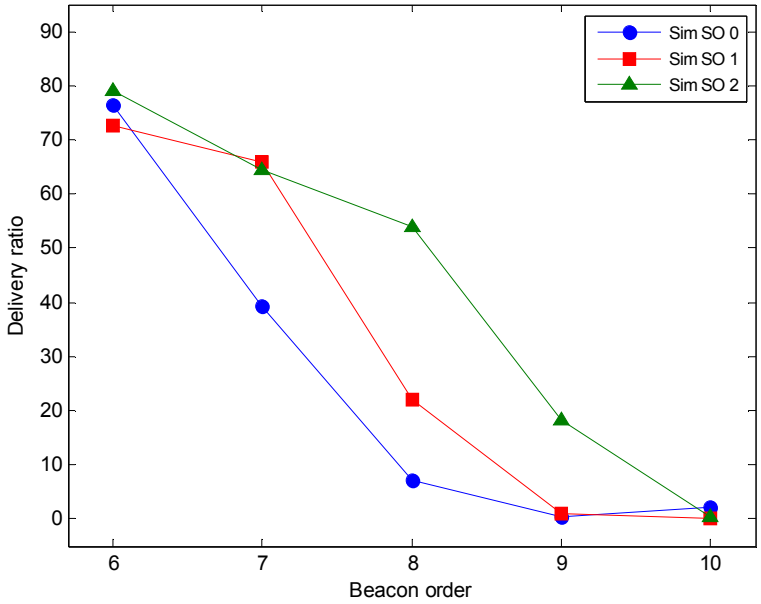


Figure 8 Average delivery ratio of nodes at data rate 30 seconds/packet with superframe orders 0 to 2

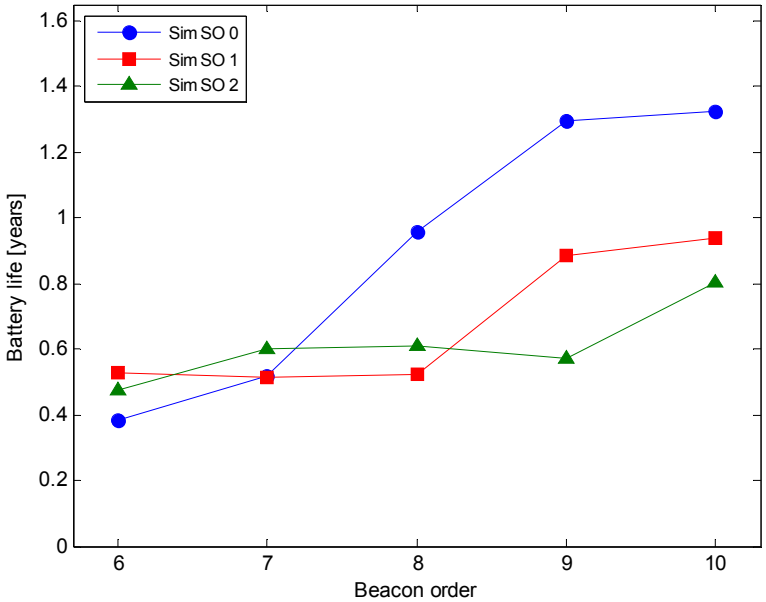


Figure 9 Average battery life of nodes at data rate 30 seconds/packet at superframe orders 0 to 2

Results at 60 seconds per packet

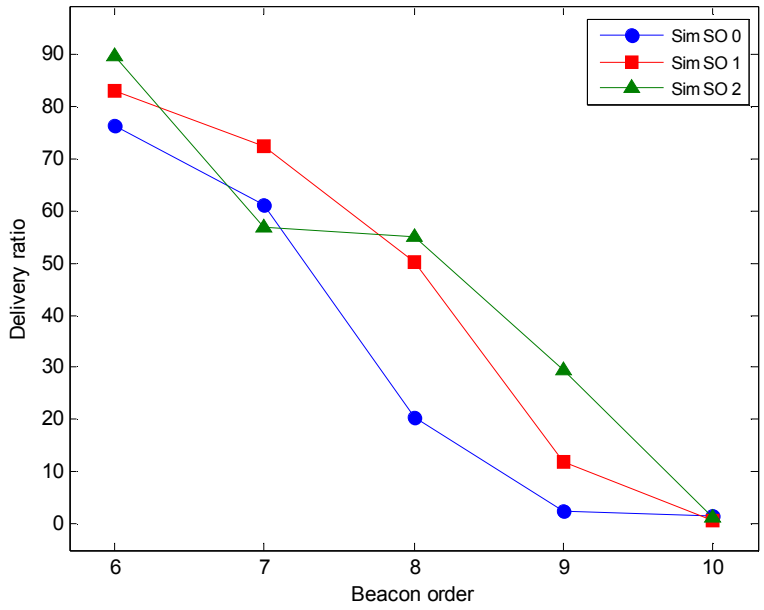


Figure 10 Average delivery ratio of nodes at data rate 60 seconds/packet at superframe orders 0 to 2

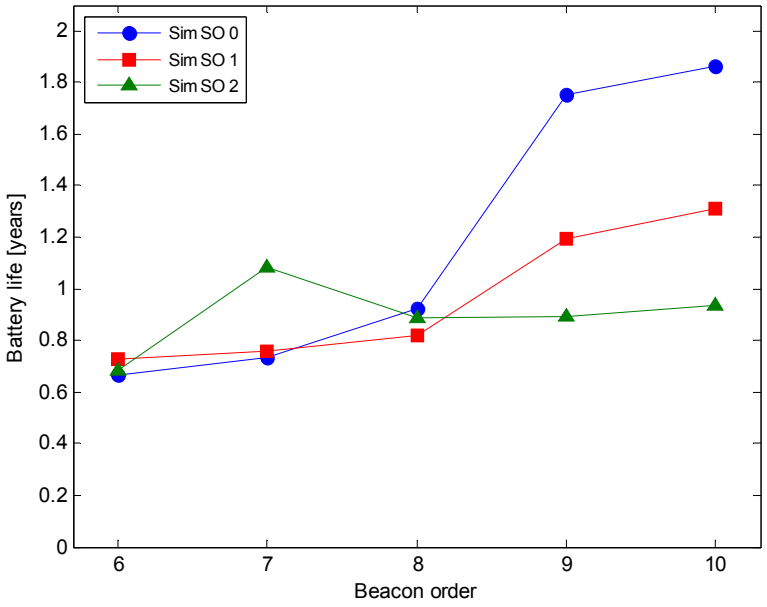


Figure 11 Average battery life of nodes at data rate 60 seconds/packet at superframe orders 0 to 2

Results at 100 seconds per packet

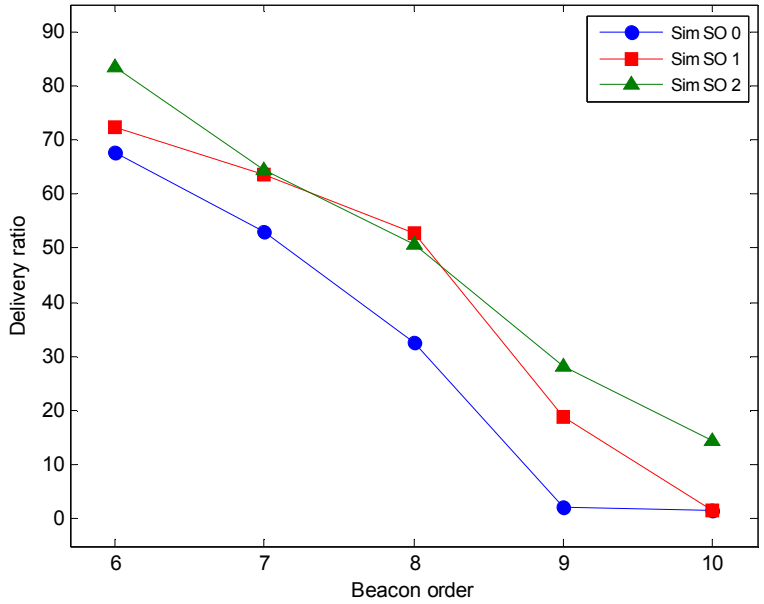


Figure 12 Average delivery ratio of nodes at data rate 100 seconds/packet at superframe orders 0 to 2

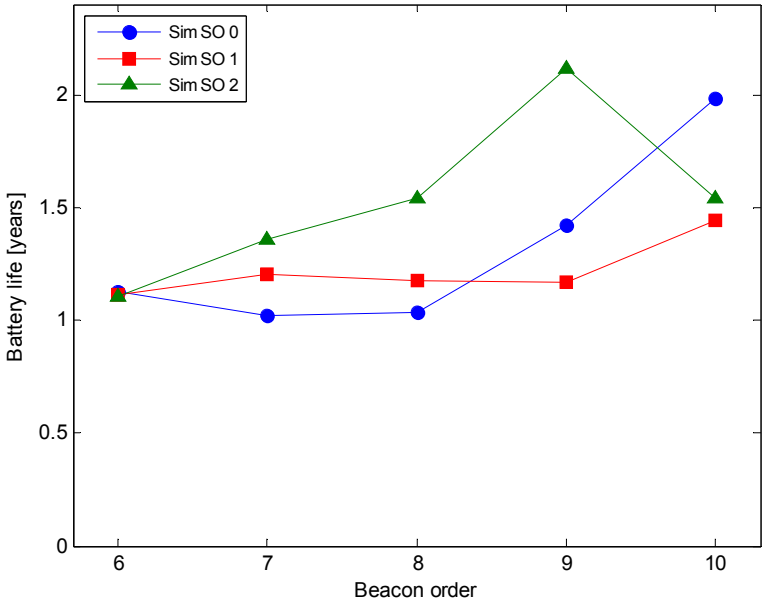


Figure 13 Average battery life of nodes at data rate 100 seconds/packet at superframe orders 0 to 2

Results at 200 seconds per packet

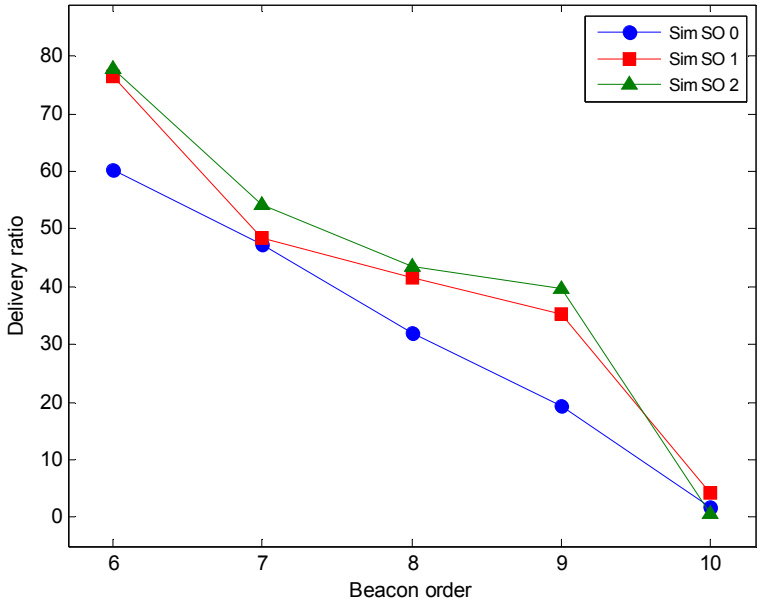


Figure 14 Average delivery ratio of nodes at data rate 200 seconds/packet at superframe orders 0 to 2

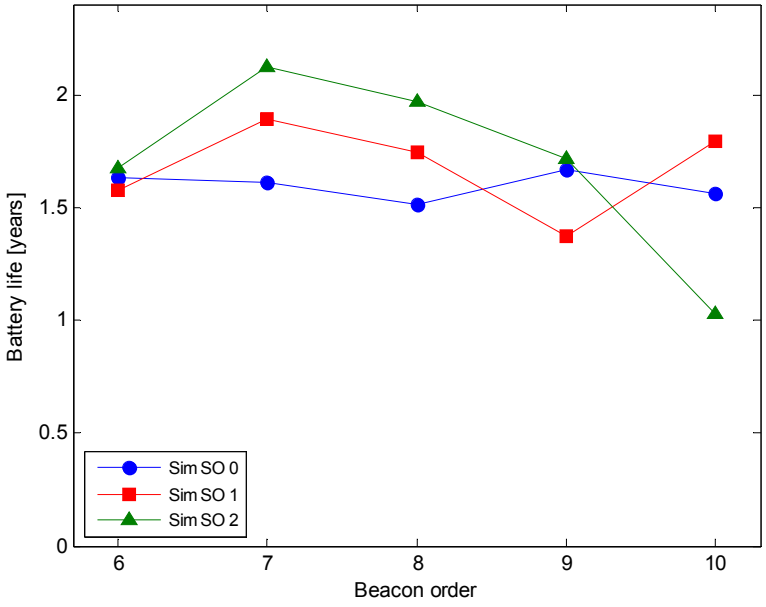


Figure 15 Average battery life of nodes at data rate 200 seconds/packet at superframe orders 0 to 2

Results at 1000 seconds per packet

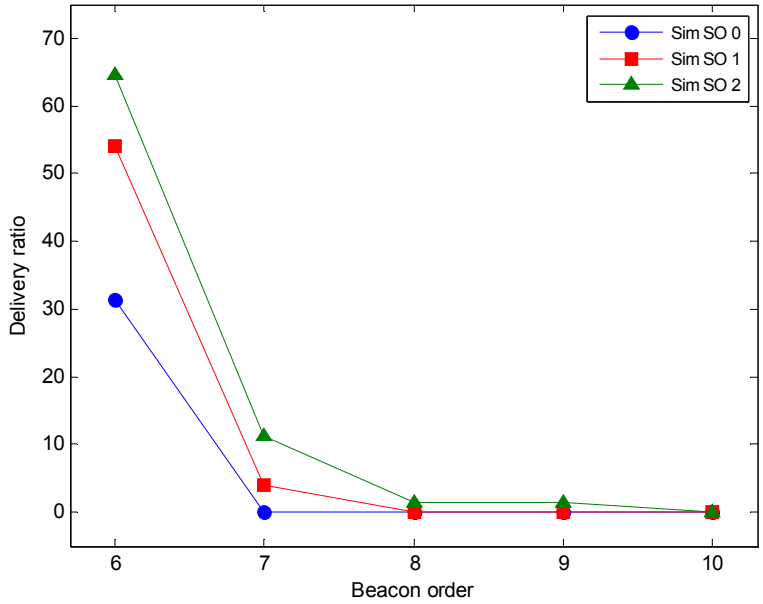


Figure 16 Average delivery ratio of nodes at data rate 1000 seconds/packet at superframe orders 0 to 2

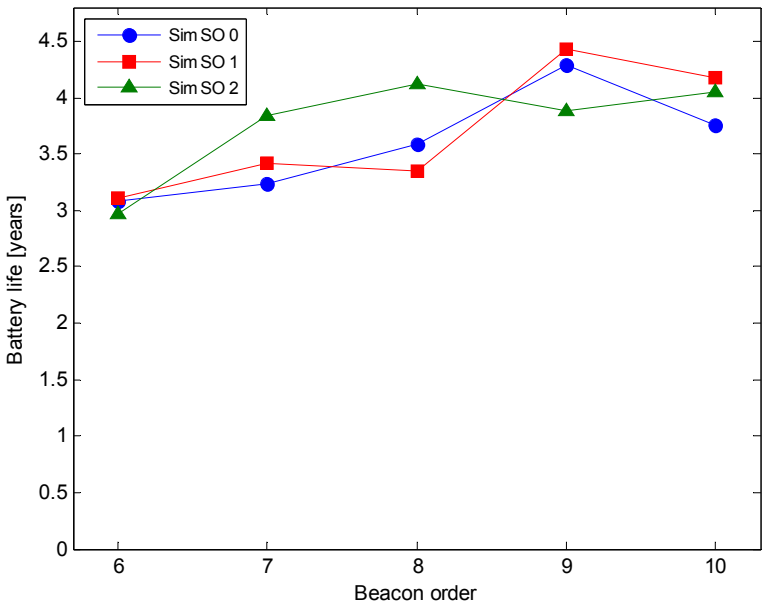


Figure 17 Average battery life of nodes at data rate 1000 seconds/packet at superframe orders 0 to 2

At all tested data rates, the delivery ratio as illustrated in Figures 8, 10, 12, 14 and 16 show the general trend of the delivery ratio decreasing as BO is increased. This is because there are fewer beacons and subsequently fewer CAPs for nodes to utilise. Therefore if a node has a collision it has fewer opportunities to retransmit.

These figures also show that on average increasing the superframe order increases the delivery ratio which was expected as this lengthens the CAP.

The results in Figures 9, 11, 13, 15 and 17 show that increasing the beacon order usually results in an increase in battery life. The exception is at 200 seconds/packet (Figure 15) where the battery life decreases as BO is increased which is completely unexpected.

3.9.2 Non Beacon Mode

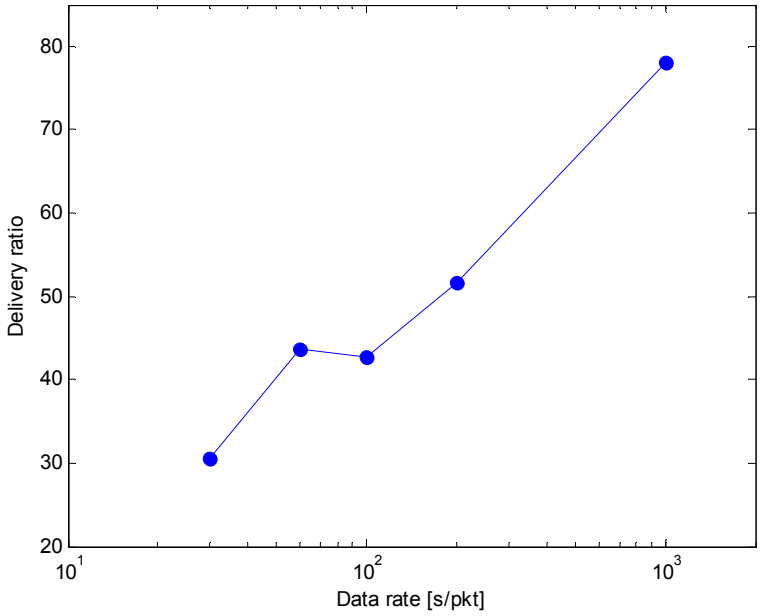


Figure 18 Average delivery ratio of nodes in non beacon mode with decreasing data rate

In non beacon mode the results are more like what would be expected. The delivery ratio improves with decreasing data rate and likewise with battery life as illustrated in Figure 18 and Figure 19. Delivery ratio is very poor however and would only just be usable at 1000 seconds per packet which is a very low data rate.

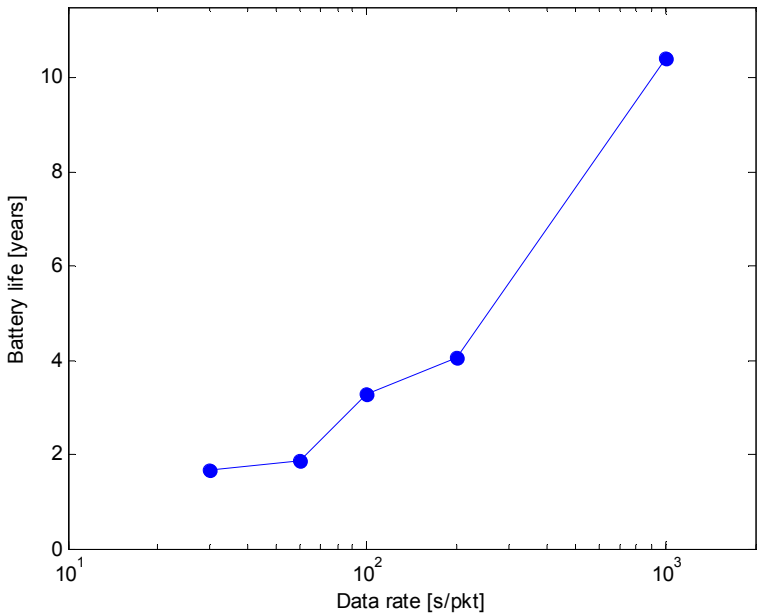


Figure 19 Average battery life of nodes in non beacon mode with decreasing data rate

3.9.3 Comments on Results

Overall the results obtained were unsatisfactory and indicate serious problems with the simulator or its configuration. In beacon mode, packet delivery ratio was only just acceptable at lower beacon orders and the delivery ratio was lowest at the lowest data rate which is the opposite of what was expected. Beacon order 10 shows a delivery ratio of almost 0 for all but one data rate. It would be expected that there might be problems at higher beacon orders such as 13 or 14 with delivery ratio due to the very long sleep time compared to the active period which would require a very accurate clock with little drift. However at beacon order 10, beacons are transmitted approximately every 15 seconds which is relatively short and should not be producing such poor results. This therefore requires further investigation.

Power consumption also appears to be excessive with many simulations showing expected battery life to be under one year even though the highest data rate still results in a very low duty cycle.

In non beacon mode, the results were more as expected with the delivery ratio improving as the data rate decreases and battery life increasing too. However delivery ratio was still very

poor and even at the fastest data rate of 30 seconds per packet it was still expected to have a delivery ratio above 80 %.

Based on the above results, it is assumed that the lower than expected battery life results are due to excessive collisions, dropped packets and retransmissions. The poor delivery ratio results, especially at higher beacon orders also suggest this. Analysis of the trace files generated by NS-2 to determine if excessive collisions are occurring is discussed in the next section.

Also noted in the above results is the large number of random outlier points in the data, where the delivery ratio or battery life will suddenly increase or decrease. This appears to be a problem with the event scheduler in NS-2 because in later experiments when ad-hoc on-demand distance vector (AODV) and address resolution protocol (ARP) broadcast packets are disabled and the experiments are repeated with just one node, the results are improved considerably. These problems were discovered in later experiments and are discussed in Section 3.9.1 and Section 3.12.

It is interesting that the initial results are so poor considering that NS-2 has been used in several research papers which show good results but do not make any mention of initial problems with the simulator. It is suspected that other published work gives good results because typically beacon mode is studied which gives considerably better results than non beacon mode. Also they typically study at a higher duty cycle (higher superframe order) than this research, however doing so defeats the purpose of a low power network because of the long active period.

3.9.4 Trace File Analysis

An NS-2 trace file is a simple text file which lists simulation results in chronological order with each value separated by a white space. The format of the trace file is illustrated in Table 10. To provide further insight as to why the power consumption and delivery ratio results are not as expected, the raw NS-2 trace files were opened in a text editor for analysis. A snippet of a trace file is provided on the following page.

Trace file snippet:

```
s 94.425152000 _3_ MAC  --- 0 AODV 55 [0 ffffffff 3 800] [energy
29159.998563 ei 0.000 es 0.000 et 0.000 er 0.001] ------- [3:255 -1:255 30
0] [0x2 1 1 [0 0] [3 4]] (REQUEST)
N -t 94.425152 -n 2 -e 29159.998250
N -t 94.425152 -n 4 -e 29159.998276
N -t 94.425152 -n 0 -e 29159.998458
N -t 94.425152 -n 1 -e 29159.998328
N -t 94.425152 -n 5 -e 29159.998328
N -t 94.425152 -n 10 -e 29159.998276
N -t 94.425152 -n 6 -e 29159.998250
N -t 94.425152 -n 9 -e 29159.998354
N -t 94.425152 -n 7 -e 29159.998379
N -t 94.425152 -n 8 -e 29159.998328
D 94.427104021 _2_ MAC  APS 0 AODV 55 [0 ffffffff 3 800] [energy
29159.998250 ei 0.000 es 0.000 et 0.000 er 0.002] ------- [3:255 -1:255 30
0] [0x2 1 1 [0 0] [3 4]] (REQUEST)
D 94.427104021 _4_ MAC  APS 0 AODV 55 [0 ffffffff 3 800] [energy
29159.998276 ei 0.000 es 0.000 et 0.000 er 0.001] ------- [3:255 -1:255 30
0] [0x2 1 1 [0 0] [3 4]] (REQUEST)
D 94.427104039 _1_ MAC  APS 0 AODV 55 [0 ffffffff 3 800] [energy
29159.998328 ei 0.000 es 0.000 et 0.000 er 0.001] ------- [3:255 -1:255 30
0] [0x2 1 1 [0 0] [3 4]] (REQUEST)
D 94.427104039 _5_ MAC  APS 0 AODV 55 [0 ffffffff 3 800] [energy
29159.998328 ei 0.000 es 0.000 et 0.000 er 0.001] ------- [3:255 -1:255 30
0] [0x2 1 1 [0 0] [3 4]] (REQUEST)
D 94.427104054 _10_ MAC  APS 0 AODV 55 [0 ffffffff 3 800] [energy
29159.998276 ei 0.000 es 0.000 et 0.000 er 0.001] ------- [3:255 -1:255 30
0] [0x2 1 1 [0 0] [3 4]] (REQUEST)
```

Table 10 NS-2 trace file parameters

Position	Parameter	Description
1	Event Action	s-Sent, r-Received, D-Dropped, N-energy update
2	Time	Time when event happened
3	Node	Node where the event happened
4	Layer	Layer where event happened: AGT-application layer, RTR – routing layer, LL – link layer (ARP performed here), IFQ – Outgoing packet queue (between link and mac layer), MAC – MAC layer, PHY- physical layer
5	Reason	Reason when packet dropped
5	Node	For an energy update trace (N) this is the address of the node
6	Packet ID	Incremental packet identification
7	Packet type	cbr – Constant bit rate data stream packet, AODV – routing packet, ARP – address resolution protocol packet
8	Size	Size of packet in bytes
9	MAC duration	Packet duration in MAC layer
10	Destination	Address of destination node
11	Source	Address of source node
14	Energy	Energy remaining at node in Joules

Several important observations were made while analysing the trace files:

- There were a significant number of collisions due to large numbers of AODV routing packets being sent between nodes.
- ARP packets were also being sent in large volumes between nodes.
- When a collision occurred, it was often more than two nodes causing the collision. Sometimes all ten nodes would transmit packets at exactly the same time.

This behaviour had not been observed while monitoring traffic between the Zigbee development boards using a packet sniffer with the sample firmware provided by Chipcon.

The problem of multiple nodes colliding is of particular concern because it hints at a problem with the NS-2 event scheduler as the nodes were not configured to avoid sending packets at exactly the same time. These problems are discussed in detail in the following section.

3.10 Zigbee Simulator Model Modifications

Initial simulator results were very poor and not what was expected based on previous research and preliminary testing with the Chipcon development boards. Since the 802.15.4 model in NS-2 has been used in many research applications it was a surprise that the results were very different. After repeating the hardware experiments a number of times and ending up with similar results it was assumed that either there was a bug in the simulator or it was incorrectly setup. The modified source code files described in the following sections can be found in Appendix D.

A detailed investigation of several source code files revealed the following problems.

3.10.1 Current Consumption Values

Performing preliminary current measurements using the development boards showed there to be a discrepancy between the transmit and receive current measurements and what was listed in the Chipcon data sheet. At a power supply of 3 volts, the same as the Chipcon data sheet specification using the same development board, the transmit current was measured to be 30

mA and the receive current to be 33 mA using a Fluke precision ammeter with a valid calibration certificate.

Based on the current results, this adjustment will make the battery life calculation even worse than it currently is, however this change is required to provide a valid comparison between the simulations and hardware experiments.

3.10.2 AODV and ARP packets

As discussed in the above section, it was discovered there were large numbers of AODV and ARP packet transmissions which did not match up with real network observations. The simulations involved a star network with one way communication between the node and coordinator and therefore the next hop is always the destination and thus AODV routing broadcasts are not required. Given this information, NS-2 can be suitably modified to prevent the resolve procedure for route finding being called by assuming the route is always present in the routing table and filling the next hop address for the destination. The modifications to `aodv.c` and `arp.c` were performed by Rao [13] and were used in this research.

The modifications to `aodv.c` are shown below.

In the function `AODV::recv()` the following has been replaced,

```
if ( (u_int32_t)ih->daddr() != IP_BROADCAST)
   rt_resolve(p);
 else
   forward((aodv_rt_entry*) 0, p, NO_DELAY);
```

with,

```
 if ( (u_int32_t)ih->daddr() != IP_BROADCAST)
   forward((aodv_rt_entry*) 1, p, NO_DELAY);
 else
   forward((aodv_rt_entry*) 0, p, NO_DELAY);
```

This change prevents the stack from trying to resolve the destination address by assuming it is already in the routing table and thus directly forwarding the packet to the destination.

In function `AODV::forward()`, the following code has been commented out:

```
if(ih->ttl_ == 0) {

#ifdef DEBUG
  fprintf(stderr, "%s: calling drop()\n",
__PRETTY_FUNCTION__);
#endif // DEBUG

  drop(p, DROP_RTR_TTL);
  return;
 }
```

`Ih->ttt_` is the time to live of a packet. It indicates the maximum number of hops a packet can make to reach the destination. Since the next hop is always the destination this variable can be ignored.

And the following has been changed from,

```
if (rt) {
   assert(rt->rt_flags == RTF_UP);
   rt->rt_expire = CURRENT_TIME + ACTIVE_ROUTE_TIMEOUT;
   ch->next_hop_ = rt->rt_nexthop;

   ch->addr_type() = NS_AF_INET;
   ch->direction() = hdr_cmn::DOWN;       //important: change
the packet's direction
 }
```

To,

```
if (rt) {
   ch->next_hop_ = ih->daddr();

   ch->addr_type() = NS_AF_INET;
   ch->direction() = hdr_cmn::DOWN;       //important: change
the packet's direction
 }
```

This change manually substitutes the destination address of the packet into the `next_hop_` variable so that the packet is sent to the destination now that the route resolving has been disabled.

An ARP packet is generated by a node when it does not know the MAC address of the next hop in the network. Nodes receiving this packet reply with the MAC address of the destination node if they have it in their ARP table. Given the minimal requirements of the star topology, an ARP request message is unnecessary. This fact was also noted by [13]and [30] who also disabled ARP messages.

Normally the function `arplookup()` in `arp.c` is called when a packet is to be transmitted and this returns 0 if there is no entry for the destination. If `arplookup()` returns 0 then the function `arpresolve()` sends an ARP request message.

The following function has been modified as shown below to disable ARP broadcasts.

From,

```
ARPEntry* ARPTable::arplookup(nsaddr_t dst)
{
     ARPEntry *a;

     for(a = arphead_.lh_first; a; a = a->nextarp()) {
          if(a->ipaddr_ == dst)
               return a;
     }
     return 0;
}
```

To,

```
ARPEntry* ARPTable::arplookup(nsaddr_t dst)
{
     ARPEntry *a;

     for(a = arphead_.lh_first; a; a = a->nextarp()) {
          if(a->ipaddr_ == dst)
               return a;
     }
     a = new ARPEntry(&arphead_, dst);
     a->up_ = 1;
     a->macaddr_ = dst;
     return a;
}
```

This change adds a new ARP entry into the lookup table if the destination address has not been found already. The destination address is manually added to the address field of the ARP table entry.

3.11 Results Without AODV and ARP

The simulator was modified to incorporate the changes described in the previous section and now contained the following modifications:

- AODV routing disabled
- ARP broadcasts disabled
- Current consumption changed to reflect results obtained from experimental testing

The simulator was then recompiled and the same simulations shown in the previous section were rerun through the simulator to validate the changes. The results are shown below.

3.11.1 Beacon Mode

This section contains the battery life and delivery ratio results for beacon mode operation at data rates 30, 60, 100, 200 and 1000 seconds/packet which are illustrated in Figure 20 to Figure 29 on the following pages. It was decided to exclude the power consumption graphs from this section since they are just the inverse of the battery life and for this research, an estimate of battery life was a more meaningful metric. However the power consumption results are available in Appendix B.

Results at 30 seconds per packet

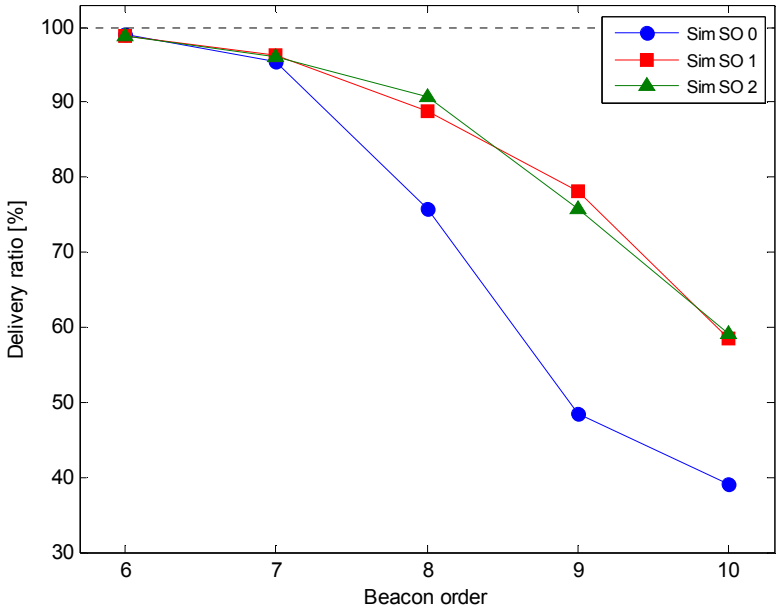


Figure 20 Average delivery ratio of nodes at data rate 30 seconds/packet at superframe orders 0 to 2

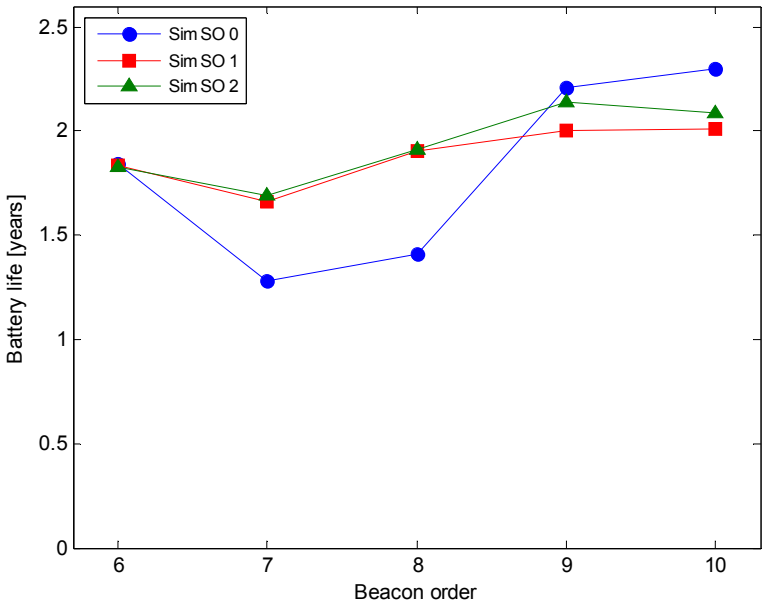


Figure 21 Average battery life of nodes at data rate 30 seconds/packet at superframe orders 0 to 2

Results at 60 seconds per packet

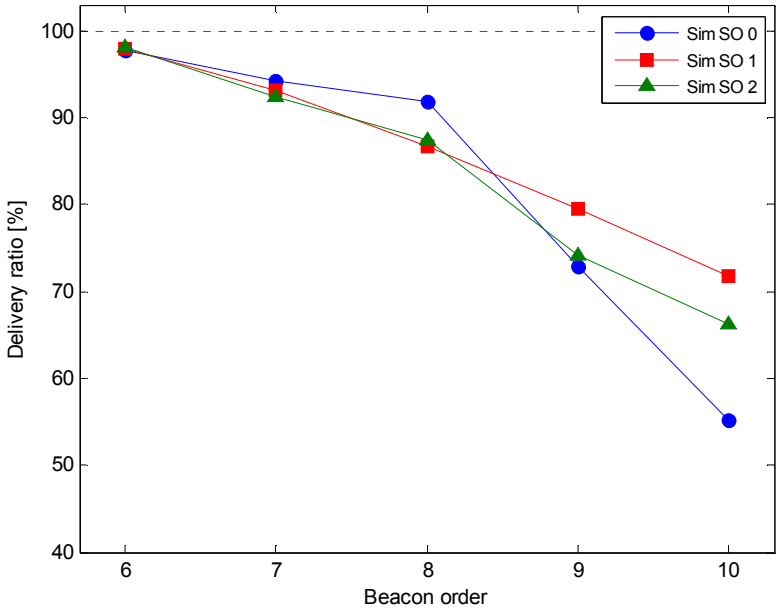


Figure 22 Average delivery ratio of nodes at data rate 60 seconds/packet at superframe orders 0 to 2

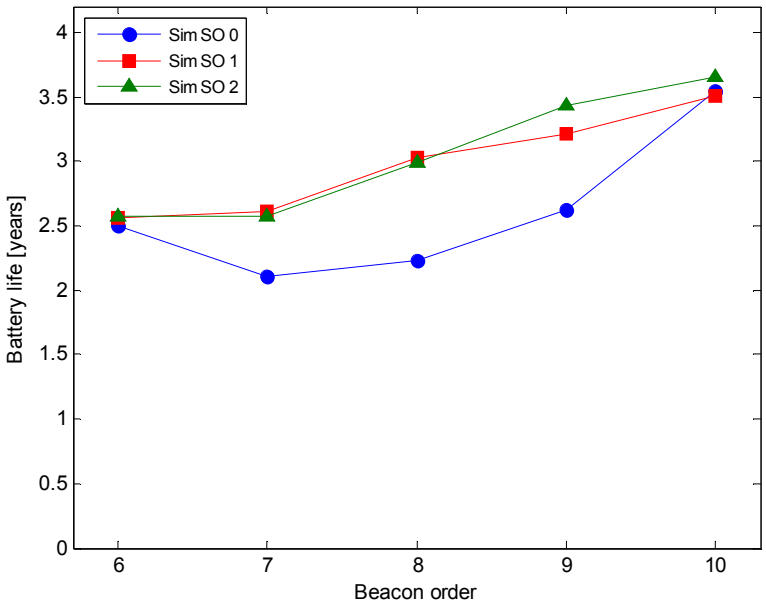


Figure 23 Average battery life of nodes at data rate 60 seconds/packet at superframe orders 0 to 2

Results at 100 seconds per packet

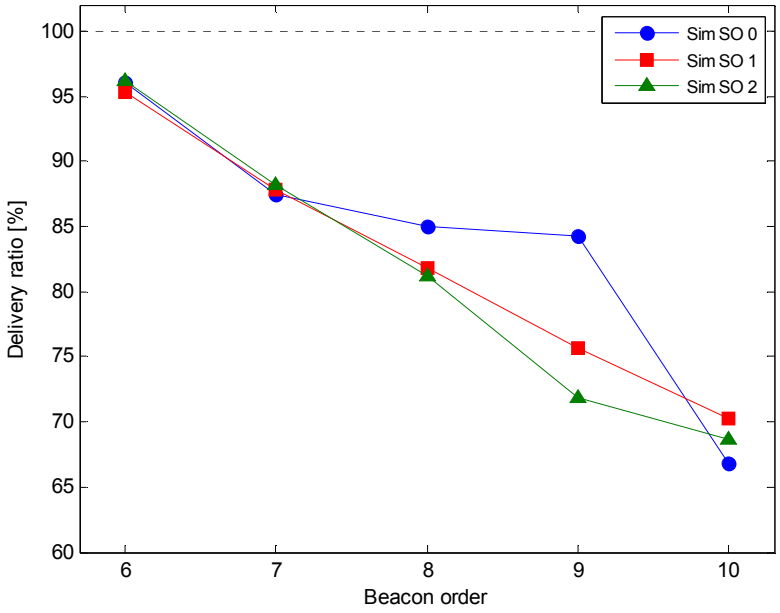


Figure 24 Average delivery ratio of nodes at data rate 100 seconds/packet at superframe orders 0 to 2

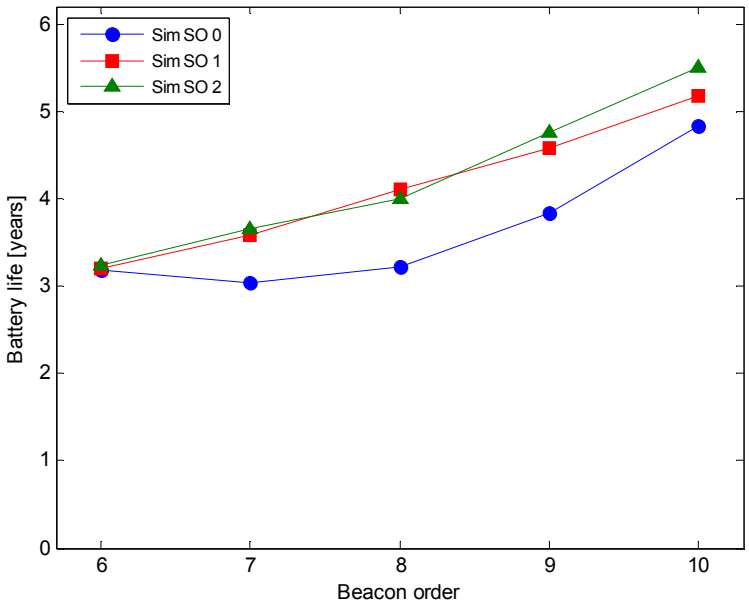


Figure 25 Average battery life of nodes at data rate 100 seconds/packet at superframe orders 0 to 2

Results at 200 seconds per packet

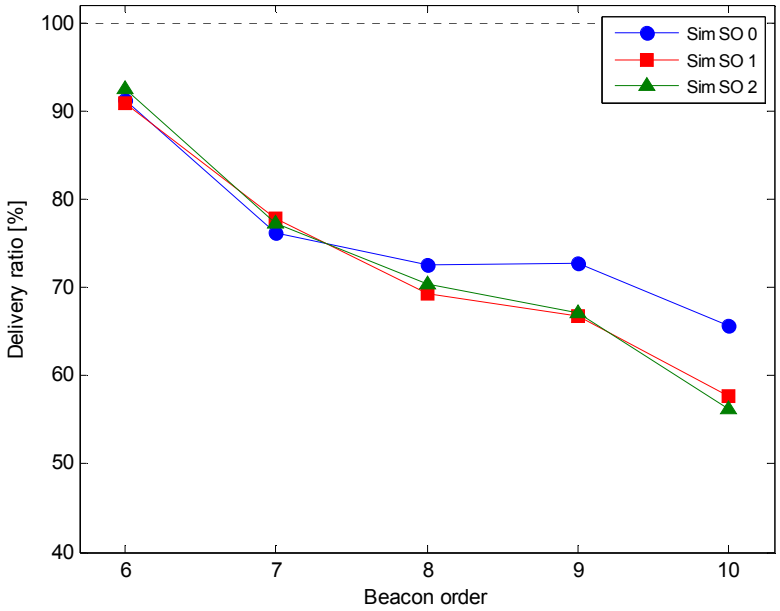


Figure 26 Average delivery ratio of nodes at data rate 200 seconds/packet at superframe orders 0 to 2

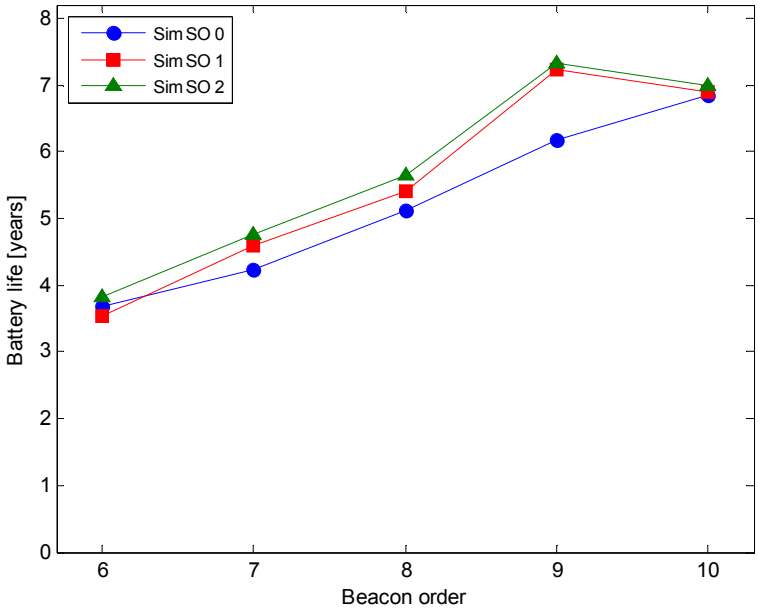


Figure 27 Average battery life of nodes at data rate 200 seconds/packet at superframe orders 0 to 2

Results at 1000 seconds per packet

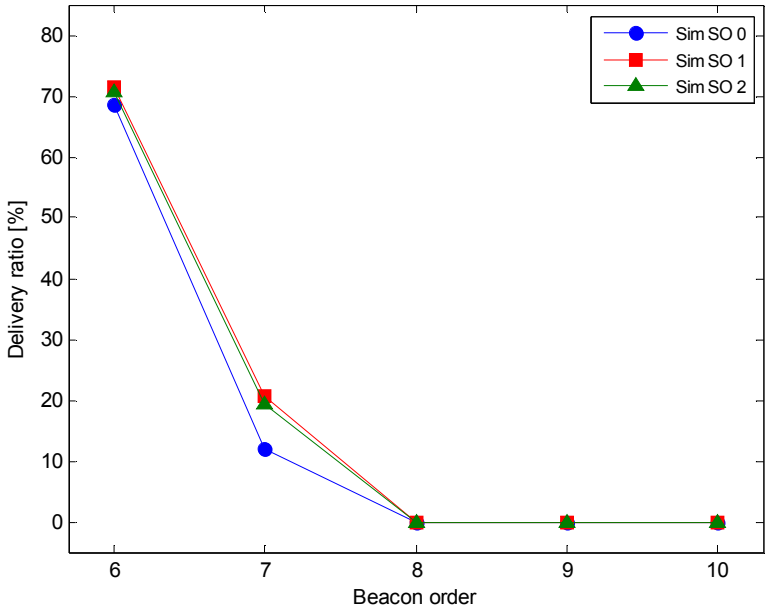


Figure 28 Average delivery ratio of nodes at data rate 1000 seconds/packet at superframe orders 0 to 2

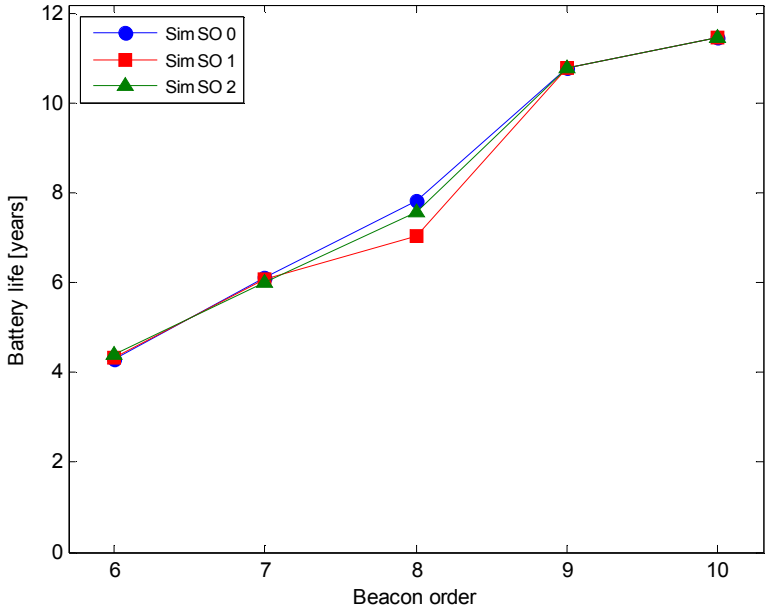


Figure 29 Average battery life of nodes at data rate 1000 seconds/packet at superframe orders 0 to 2

As seen in the above results, preventing the extra ADOV and ARP packet broadcasts has considerably improved the performance of the nodes with both the delivery ratio and battery life improving.

Like the initial simulation results, the delivery ratio results in Figures 20, 22, 24, 26 and 28 decreases with increasing BO. Figures 21, 23, 25, 27 and 29 show a much greater increase in battery life as BO increases compared to the initial results. These results are further discussed at the end of this section.

3.11.2 Non Beacon Mode

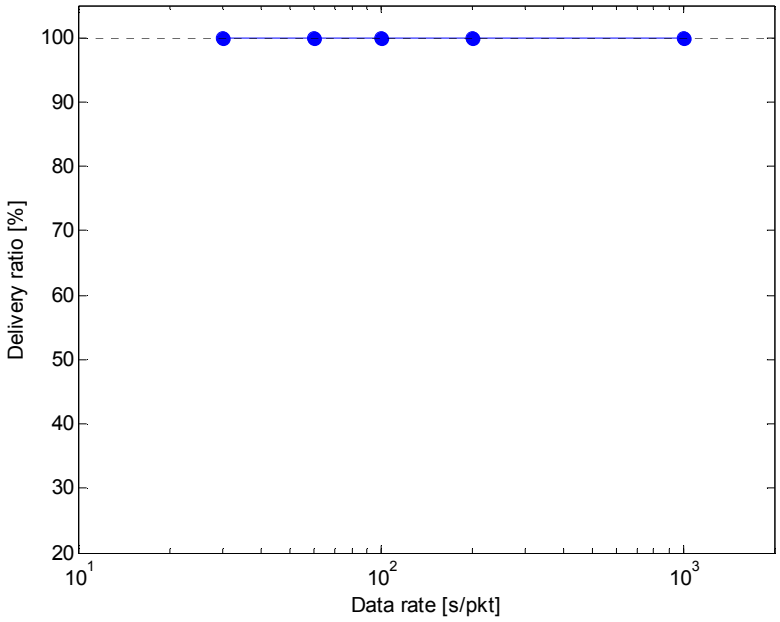


Figure 30 Average delivery ratio of nodes in non beacon mode with decreasing data rate

Removing AODV and ARP has dramatically changed the non beacon results. Figure 30 shows that no packets are lost at any data rate with very good battery life as illustrated in Figure 31.

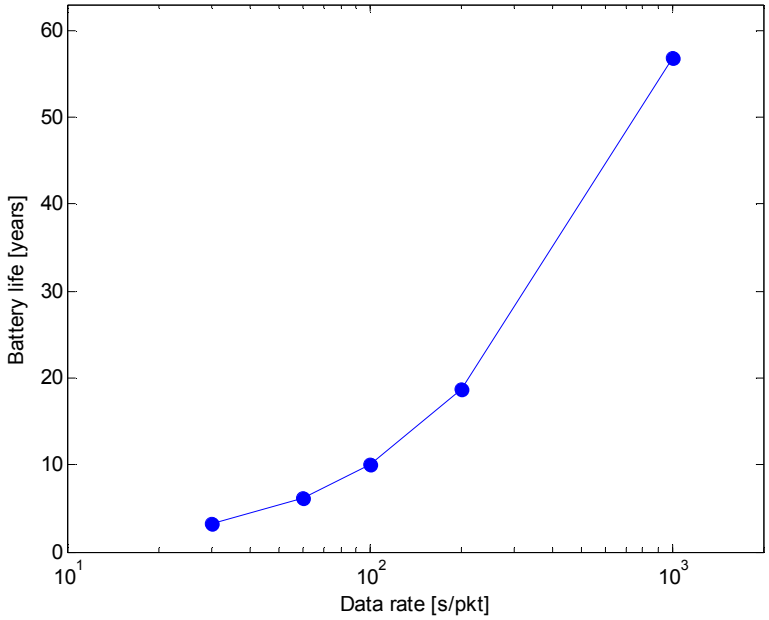


Figure 31 Average battery life of nodes in non beacon mode with decreasing data rate

3.11.3 Comments on Results

The results obtained from the modified simulator model are a significant improvement. Delivery ratio was much improved and was usually above 80 percent which is what was expected. Battery life also increased considerably.

These results show an interesting trend in beacon mode where superframe order 0 frequently has a shorter battery life than superframe order one or two despite the longer active period. This trend was also noted by Kohvakka *et al*. [6] however it is much more noticeable in the above results. It is due to increased contention and retransmission for the channel as the active period where nodes can transmit is shorter. If a device trying to transmit senses the channel to be busy it then waits a random amount of time before retrying the transmission. This delay causes the increase in power consumption and subsequently a shorter battery life due to the extra power consumed while sensing the channel and subsequently re-sensing the channel before transmission. Higher superframe orders have a longer active period therefore there is less contention for the channel. It was not expected that there would be such a significant difference between superframe order 0 and superframe order 1 or 2 because even the fastest data rate of 30 seconds/packet has a very low duty cycle and the nodes are not configured to send packets at exactly the same time.

Delivery ratio for the lowest data rate of 1000 seconds/packet was still not what was expected, with the delivery ratio falling to zero at beacon order 8. Since the simulation time is 1000 seconds each node only transmits one data packet during the simulation, if it fails then its delivery ratio will be zero. However the delivery ratio is an average of all 10 nodes and it would be unlikely that all 10 nodes would be unable to send data to the coordinator. Further investigation is required to determine where the packets are being dropped.

In non beacon mode the results are much improved with 100 percent delivery achieved for all data rates and battery life has increased five times from the initial simulations.

3.12 Single Node

While performing preliminary simulations it was noted that the simulator gave quite different results when there was only a single node, rather than 10 nodes. Delivery ratio and battery life were improved considerably and appeared to follow the experimental results much more closely. It was also noted during the experimental investigation that at the data rates being tested, there appeared to be minimal difference in battery life and delivery ratio between one node and 10 nodes. Because of this, the final results were generated using a single node in the simulation and hardware experiments.

Other researchers [4, 5] have shown that node power consumption increases as the number of nodes in the network increases, however initial experimental testing has shown that there is little difference between 1 and 10 nodes. This does, however highlight another issue with the simulator which needs resolving.

3.13 State Transitions

The original wireless NS-2 energy model did not take state transitional energy consumption into account. Code contributions by Ramachandran [22] added a fixed time delay to each transition to take the delay of switching from transmit to receive mode into account.

According to the Chipcon CC2430 data sheet[31], the transition time was correct, however the data sheet shows that extra transition time is required when initially powering up the transceiver and powering it down. This happens frequently in Zigbee end devices as devices try to remain in sleep mode as much as possible and only power up when necessary. This extra time interval had not been taken into account in the energy model and it was subsequently modified to reflect the case. A more satisfactory solution would be to add these transition times as TCL variables which could allow these time constants to be changed from the simulation script, however it was much simpler to modify them directly and it would have required a moderate amount of time to add the TCL commands.

Chapter 4 – Data Acquisition

4.1 Introduction

This section details the procedures, equipment and setup required for characterising power consumption and delivery ratio in a Zigbee wireless network and to validate the accuracy of the NS-2 network simulator for Zigbee wireless networks. These experiments involve measuring two metrics simultaneously, power consumption and delivery ratio versus varying network and traffic parameters. The methods used in measuring these metrics are detailed in the following sections. The results are then compared with those obtained from the simulations in Chapter 3.

4.2 Methodology

A key component of this research is to validate computer simulation of a Zigbee network against a real working network. To enable comparison between the energy consumption of the simulations and the hardware development boards, a method of measuring energy consumption and delivery ratio was needed. The method chosen would need to be able to record data for 1000 seconds which is the duration of the simulations.

A decision was made to measure power consumption by measuring the voltage drop across a precision resistor using a digital storage oscilloscope and calculating the expected battery life from the results. To measure delivery ratio, a computer is connected to the coordinator which in turn forwards received packets to the computer. An application running on the computer tallies the packets and after 1000 seconds has elapsed generates a report of the results.

4.3 Equipment

This section discusses the equipment used in the experimental investigation with Table 11 listing the equipment used.

Table 11 List of equipment required for experiments

Description	Purpose	Identification details
Chipcon development kit	Contains 10 Zigbee devices including one with external serial port to create experimental Zigbee network. Also includes a Zigbee packet sniffer	CC2431DK
Laboratory power supply	Produces a stable 3 Volt supply to power the end device being monitored	Powertech MP-3086
Digital storage oscilloscope	Used to empirically determine sampling parameters to configure the data acquisition module	Tektronix TDS-1002
Data acquisition module	Records the voltage drop across the precision resistor and sends the raw data to a computer	Measurement Computing USB-1608FS
MATLAB software with Data Acquisition Toolbox	Controls the data acquisition module, stores the raw data to file, performs post processing and analysis on the data.	Version 2008a
Precision Resistor	Precision 0.1 percent resistor used to measure the current being drawn by the device under test	10 Ohm 0.1% 1 Watt resistor
Zigbee Stack and Firmware	Zigbee software for use on the Chipcon microcontrollers	Z-Stack 1.4.3
Delivery ratio application	An application written in Borland C++ Builder to measure the packet delivery ratio during the experiment and save the results to file	Version 1.0
Personal Computer	Running MATLAB to process and store the recorded data from data acquisition module and to run the C++ application	Windows XP

4.3.1 Chipcon Development Kit

Chipcon make several Zigbee development kits which were considered for this research, they are all basically similar apart from the number of devices that they contain. The CC2431 kits contain the CC2431 microcontroller which is essentially the same as the CC2430 but contains a hardware location engine. When this is disabled all other characteristics are the same as the CC2430.

CC2431DK:

- 2 SmartRF motherboards
- 10 CC2431EM battery powered nodes with antennas
- Debugging interface for programming the battery powered nodes
- Cost $999 USD

CC2431ZDK:

- Contains the same hardware as the CC2431DK but also contains 5 CC2430DB demonstration boards
- Cost $1999 USD

CC2430ZDK:

- 2 SmartRF motherboards
- 5 CC2430DB battery powered demonstration boards
- Cost $1499 USD

CC2430DK

- 2 SmartRF motherboards
- 2 CC2430DB battery powered demonstration boards
- Cost $540 USD

The particular Chipcon development kit chosen was the CC2431DK. This development kit was ideal for this research as it will enable all testing to be carried out without developing new hardware, it contains enough nodes to replicate the simulation setup, was reasonably priced and the serial port interface enabled easy connection to a PC for monitoring and data acquisition.

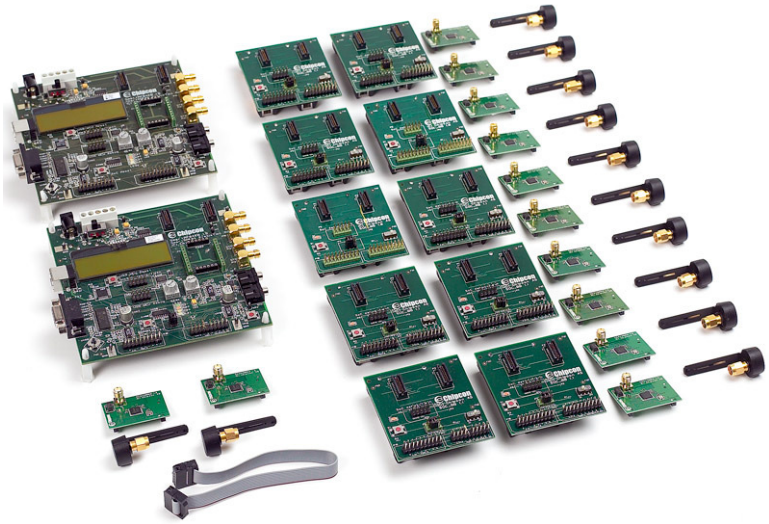


Figure 32 CC2431DK development kit [32]

4.3.2 Laboratory Power Supply

The node under test was powered from a 3 volt laboratory power supply. This was because the current consumption of the CC2430 microcontroller varies with supply voltage and powering the device from a battery would have introduced an additional variable into the results which is not present in the simulations. The Powertech MP-3086 laboratory power supply was utilised because it was on hand and was adequate for this purpose.

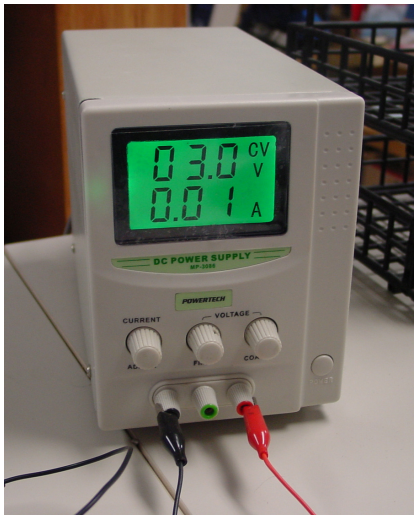


Figure 33 Laboratory power supply

4.3.3 Digital Storage Oscilloscope

The Tektronix TDS-1002 oscilloscope was used for two main tasks. It was initially used to configure the data acquisition module by viewing the voltage across the precision resistor. This enabled the trigger voltage and sample rate to be determined for the data acquisition module without having to determine them empirically. It was also used for validation of samples from the data acquisition module. This particular Tektronix oscilloscope was chosen as it was already available, therefore did not have to be purchased and was more than capable of performing the tasks required.

4.3.4 Data Acquisition Module

The data acquisition module used in this research was a Measurement Computing USB-1608FS USB data acquisition board. It was chosen for the following reasons:

- 16 bit Analogue to digital converter
- 50 kilosamples per second sampling rate
- Selectable input gain
- USB interface
- Compatible with MATLAB Data Acquisition toolbox.
- Low cost
- University had prior experience with these modules

The data acquisition module is configured from a MATLAB script which uses the Data Acquisition Toolbox component of MATLAB to control the module.

Figure 34 USB data acquisition module

4.3.5 Zigbee Stack and Firmware

Chipcon supply a fully featured Zigbee stack known as Z-Stack [33] with their microcontrollers, as well it being freely available on the internet. This software is built on a real time operating system with full task management. For this research, two separate firmware projects were built, one for the coordinator and one for end devices. This is because of the different functionality required by the two devices. Both firmware projects use the same Zigbee stack but are configured in the compile options to be either coordinators or end devices.

To replicate the operation of the simulator, a transmit data task was written for the end devices. This task is executed by the task scheduler when a specified time period has elapsed, in this case the interval is either 30, 60, 100, 200 or 1000 seconds depending on the experiment being performed. The transmit data task sends a 64 byte data packet to the coordinator. To enable the delivery ratio of the network to be easily measured, a 16 bit counter variable is added to the beginning of the data packet which fills the first two bytes of the packet. The transmit data task increments this counter each time a packet is transmitted and an application running on a PC connected to the coordinator keeps track of the counter variable to determine if any packets have been dropped.

The firmware for the coordinator is based on the same Zigbee stack as the end devices but is configured to act as a coordinator. The firmware in the coordinator also has a transmit data task as well as one to manage the serial port. The transmit data task in the coordinator works differently to the task on the end device. When a packet is received from an end device a data arrival event is triggered which schedules the transmit data task. This task reformats the data packet and transmits the data to a computer over the serial port. The purpose of this task is to enable the average delivery ratio of the network to be monitored by a custom built application running on the PC which is connected to the serial port.

The serial port task is used to clock the bytes out of the universal asynchronous receiver transmitter (UART) in the microcontroller when data is being transmitted and fills a buffer with any received data so it can be processed by other tasks.

After using the stack it became apparent that although beacon mode appeared to be supported, the underlying code in the libraries provided by Chipcon did not support it, despite beacon mode being part of the Zigbee standard[9]. Because of this, the Texas instruments 802.15.4 MAC software TiMAC [34] was used which is purely the 802.15.4 MAC software stack which supports star and peer-to-peer topologies. Because the simulator only supported the 802.15.4 stack, this would provide a valid comparison between the results. However for further validation of the technology, a Zigbee wireless network would also be created using the star and mesh network topologies to provide comparison between the Zigbee network stack and the underlying 802.15.4 stack. Therefore, four firmware projects were created, one each for 802.15.4 beacon mode, 802.15.4 non beacon mode, Zigbee mesh mode and Zigbee star mode. In order to keep consistency between the different firmware projects, all use the same data transmission and time management tasks in the application layer. The firmware can be located in Appendix D.

4.3.6 Delivery ratio application

In order to easily measure the average delivery ratio of the Zigbee network, a PC application was developed in Borland C++ to download the data packets being received by the coordinator onto the PC. An array in memory keeps track of the counter bytes for each node that it transmits at the start of each packet and compares it with the last counter value which is stored in memory. If the counter value has increased by more than one then packets have been lost. A variable for each node represents the number of dropped packets and is incremented by the difference between the received counter value and the previous counter value when packets have been lost.

After 1000 seconds of packet capture the program calculates the average delivery ratio of all 10 nodes and saves the results to a text file for later analysis. A screen shot of the delivery ratio application is shown on the following page in Figure 35.

The source code for this application can be found in Appendix D.

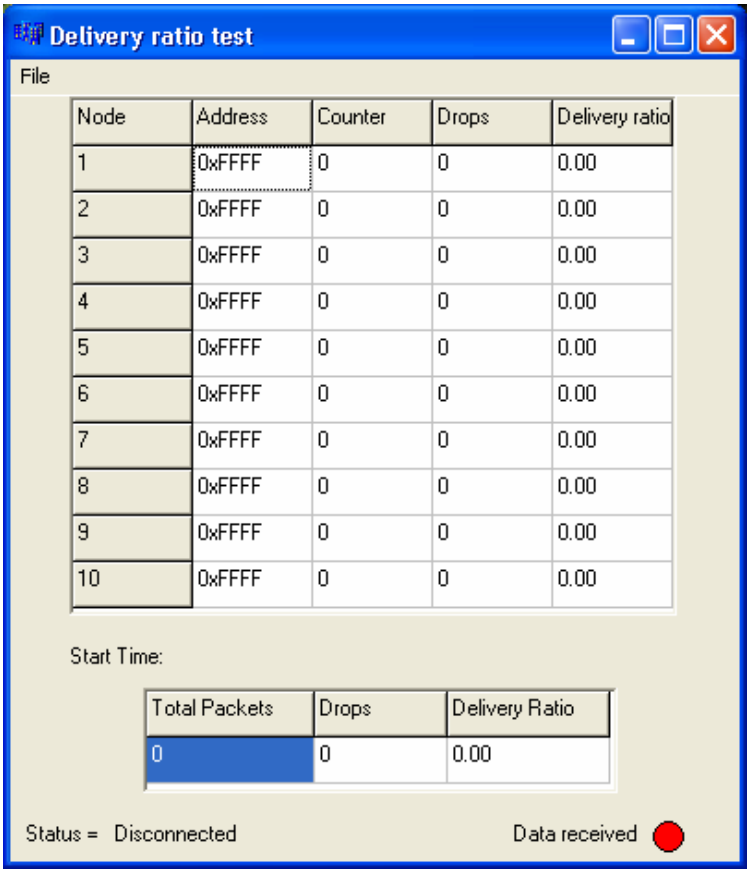


Figure 35 Delivery ratio application

4.3.7 MATLAB Software Development

In order to acquire samples from the data acquisition board a data acquisition application was required. Initial tests performed using the supplied software with the data acquisition board revealed it would only capture data for a limited number of samples. The Measurement Computing data acquisition board is also supported by the MATLAB Data Acquisition Toolbox, therefore a MATLAB script could be written to perform the data acquisition.

The Data Acquisition Toolbox provided in MATLAB provided a very flexible environment for configuring the data acquisition module. For the experiments the data acquisition module was configured with the parameters detailed in Table 12 while the MATLAB files can be located in Appendix D.

Table 12 Data acquisition board sampling parameters

Parameter	Value
Sample rate	25 kilosamples/second
Trigger Type	Software
Trigger condition	Rising
Trigger condition value	0.13 V
Samples per trigger	2500

The parameters for the data acquisition module were determined empirically by performing a number of experiments using the oscilloscope as a reference to verify the acquired signals integrity. From initial measurements with the oscilloscope it was determined that a sample rate of around 20 kilosamples per second would be sufficient to measure all features of the signal. It was decided to set the sample rate at 25 kilosamples to be sure there was sufficient detail in the acquired signal.

Initially the data acquisition module was configured to sample continuously for the duration of the experiment, however this generated an extremely large amount of data which the computer struggled with when trying to perform post processing and frequently failed with out of memory errors. After observing the active current consumption for one packet transmission on the oscilloscope it was determined that it was not necessary to sample continuously as the idle current consumption of the device remains at a reasonably constant 0.5µA. The oscilloscope also indicated that performing 2500 samples after the trigger condition was more than sufficient to capture the data of a packet transmission.

4.4 Experiment Procedure

The experimental procedure consisted of a series of trial experiments to verify the correct operation of the network and monitoring apparatus. Once the correct operation of the network had been verified, the scenarios used in the simulator could be replicated.

The scenarios to be tested involved the following parameters; these are the same as used in the simulations:

Table 13 Parameters used in the experiments

Parameter	Range
Beacon Order (BO)	6-10 and 15 (Non beacon mode)
Superframe Order (SO)	0-2 and 15 (Non beacon mode)
Data rate	30, 60, 100, 200 and 1000 seconds per packet
Battery capacity	900 mAH

As well as the 802.15.4 MAC testing, Zigbee mesh and star network topologies were also tested even though they could not be simulated with NS-2. The network topologies were arranged as shown in Figure 37 with the SMARTRF development board acting as the coordinator and the CC2431BB battery boards acting as end devices. For the Zigbee mesh network testing a CC2430DB demonstration board was programmed as the router. Figure 36 shows the individual development boards used in the experiments.

Each experiment consisted of three general steps

- Preparing equipment for an experiment
- Running the experiment
- Processing the data

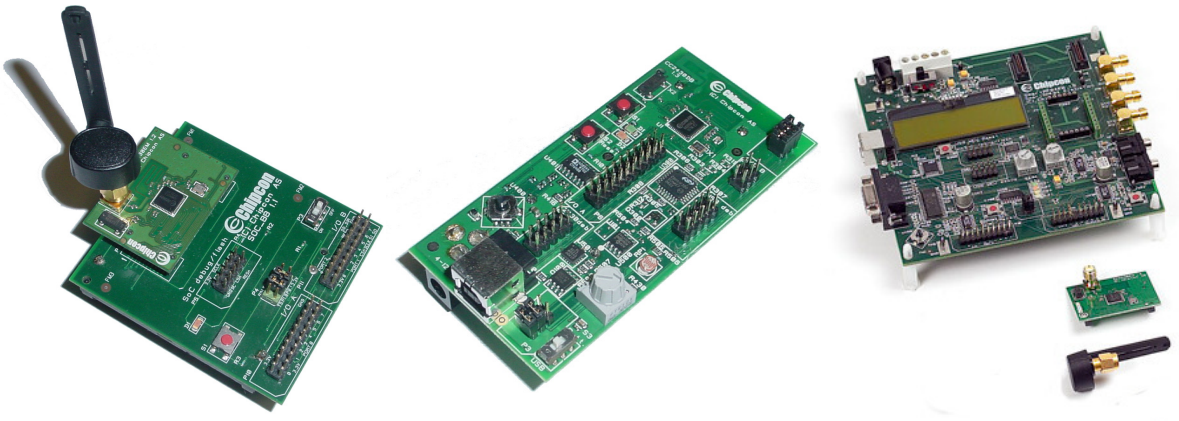


Figure 36 CC2431BB, CC2430DB and SMARTRF development boards used in the experiments

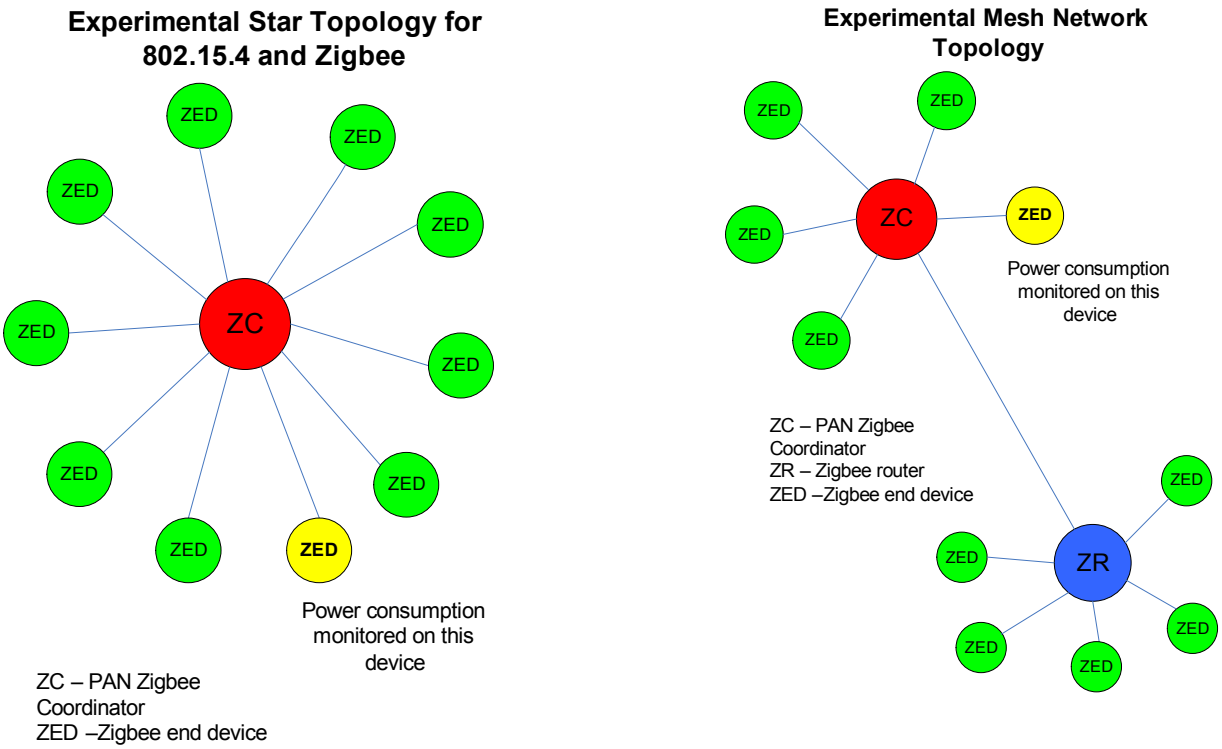


Figure 37 Network topologies as used in the experimental investigation

4.4.1 Preparing Equipment for an Experiment

Before each experiment could begin the equipment used in the experiment needed to be validated and configured for the scenario being tested.

- The firmware running on the devices needed to be configured with the required beacon order, superframe order and data rate being tested
- The firmware is then compiled and programmed into each device being used
- The oscilloscope is used to check the supply voltage of the power supply to make sure it is providing a stable 3 volts.
- The data acquisition module is checked to make sure it is reading 0 volts across the resistor with no current flowing through it
- The MATLAB script is configured for the data rate being tested.
- The Zigbee packet sniffer log file was erased, ready for the next experiment

Finally the equipment was connected as illustrated in Figure 38 and a photo of the experiment operating can be seen in the following section in Figure 39.

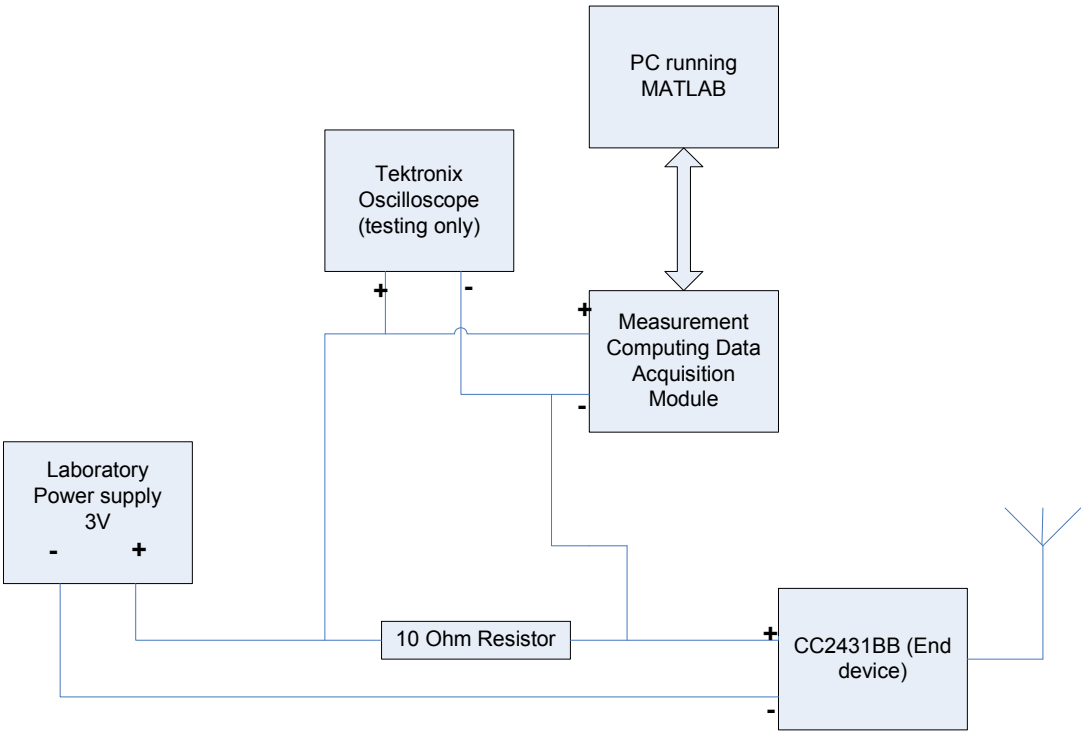


Figure 38 Schematic diagram of power measuring setup

4.4.2 Running the Experiment

The first step was to start the Zigbee packet sniffer, followed by switching on the coordinator. The status of the coordinator is displayed on an LCD screen on the development board which displayed when it had created the network along with the personal area network identifier (PAN-ID). The status of the network can also be seen in the packet sniffer log file.

Once the coordinator had started and initialised the network the end devices were powered on. In the computer simulations, there is a five second gap between each end device being started; therefore each device was switched on five seconds after the previous device. Lastly the end device under test is powered on via the laboratory power supply. Once switched on, the end devices associate with the coordinator and then begin sending packets at the programmed rate. Figure 39 shows the node under test with the experiment underway.

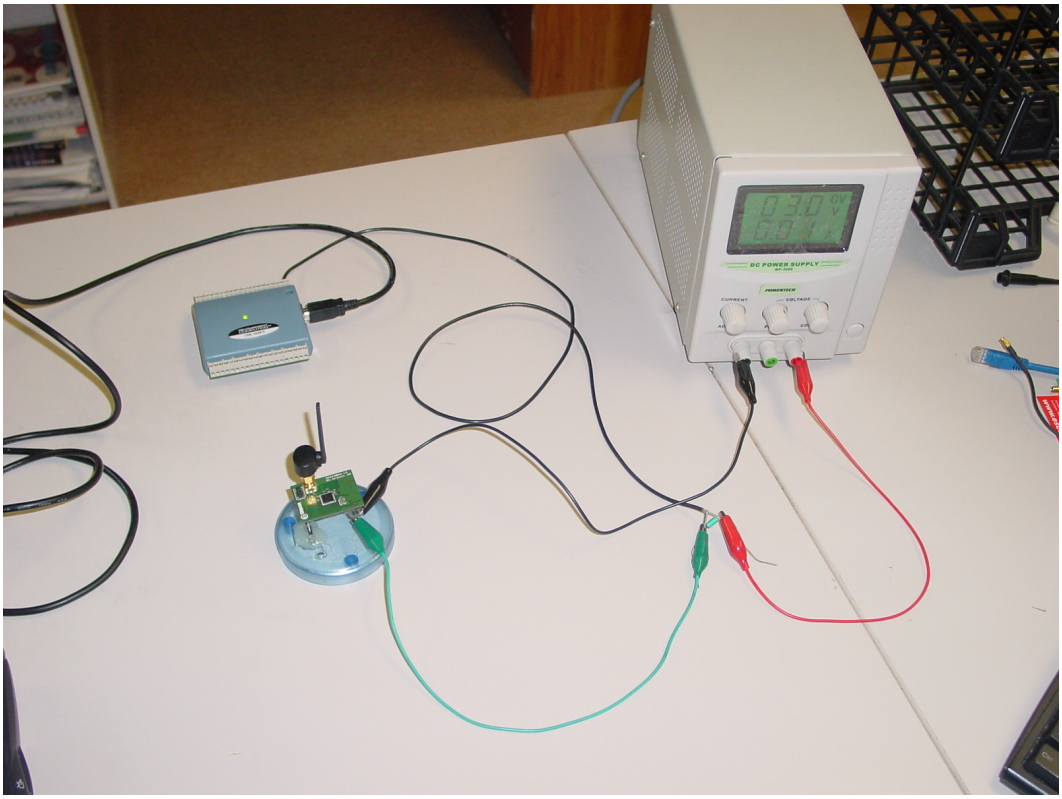


Figure 39 Experiment setup for measuring current consumption of network node

When the end device under test has associated with the coordinator and begun sending packets the data acquisition was started by running the MATLAB script from the computer and the next 1000 seconds of current consumption was recorded. Due to the amount of data being processed, the MATLAB script streams the logged data directly to a file on disk for later processing.

While the experiment is underway, an application running on the PC is also recording the delivery ratio of the data transmission between the coordinator and the end devices in the network.

Once the required number of samples has been taken the MATLAB script then loads the file for post processing and the PC delivery ratio application saves the delivery ratio results to a text file for later analysis.

4.4.3 Processing the Data

The aim of performing the data acquisition is to determine the battery life of the end device by measuring energy consumption as well as the delivery ratio of the network. The delivery ratio results needed no further processing and could be imported directly into Microsoft Excel for analysis and graphing. The energy consumption results however require further conversion to calculate the expected battery life.

Several steps are required to calculate the expected battery life. Firstly the raw data is loaded from the file on disk into two arrays, one representing the time elapsed and the other containing the raw data in volts.

The second step is to determine the total active time and total inactive time of the end device. When the voltage across the sense resistor rises above 0.13 volts the data acquisition board triggers and takes 2500 samples to capture the active mode current consumption of the end device.

Each time the data acquisition board triggers it saves a Not a Number (NaN) value into the data array. The script scans the data array and calculates the time period between the last sample in the previous trigger period and NaN for the current trigger period for each sample period in the array. The sum of the calculated time periods gives the total non-sampled time when the end device was in sleep mode. It is assumed that the current consumption is a constant 0.5μA for the end device during the non-sampled time, which is fairly accurate based on a number of experiments.

The next step is to convert the raw data from volts to amperes by dividing the voltage by the resistance of the sense resistor, which in these experiments is 10 ohms.

The next task is to sum the current measurements to determine the total charge consumed when the device is active by multiplying by the time spent in active mode. The charge consumed while the data acquisition board was not sampling is determined by multiplying the sleep mode current consumption of 0.5μA by the total non-sampled time. The total

charge consumed is the calculated by adding the active mode charge with the calculated non-sampled charge.

The total energy consumed is then calculated by multiplying the total charge by the inverse of the total experiment time. Finally the battery life is calculated by dividing the ampere-hour capacity of the battery by the total energy consumption and converting the result from seconds to years.

Initial results from the experiments gave significantly different results to the computer simulations, with the battery life of the simulation significantly greater than the experiment results. After observing the acquired data it became apparent that there was a moderate amount of CPU activity before the actual packet transmission which the simulator is not able to take account of. Therefore it was decided to strip the CPU power consumption data from the results to see if it improved the results. This brought the battery life much closer to the simulation results, therefore code was added into the MATLAB script to remove the CPU current consumption to make fair comparison between the results. The investigation into this problem is discussed later in Section 4.6.

In summary, the energy consumption data acquisition analysis was as follows

- Acquire 1000 seconds of data from data acquisition board
- Convert voltage measurements to current
- Calculate total time between each trigger period
- Remove CPU current consumption samples
- Calculate total charge consumed from acquired data and charge consumed while idle
- Calculate total battery life

4.5 Preliminary Results

The test equipment was configured as described above and each of the parameters was tested following the above test procedure which produced the results described in the following sections.

4.5.1 Beacon Mode

Results at 30 seconds per packet

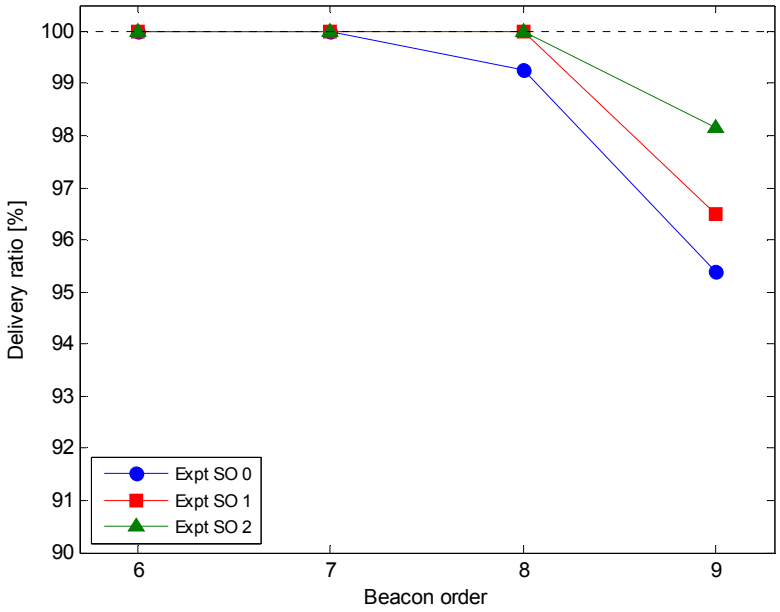


Figure 40 Average delivery ratio of nodes at 30 seconds/packet at superframe orders 0 to 2

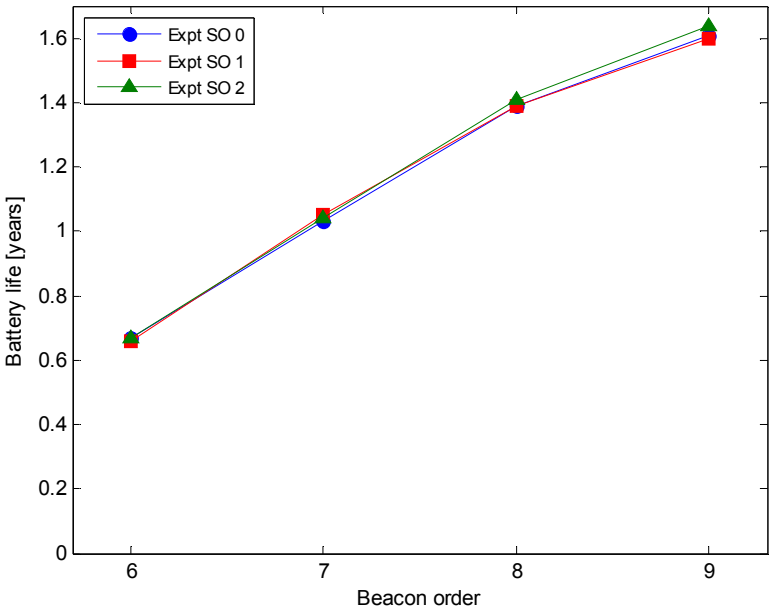


Figure 41 Battery life of node with increasing beacon and superframe orders at data rate 30 seconds/packet

Results at 60 seconds per packet

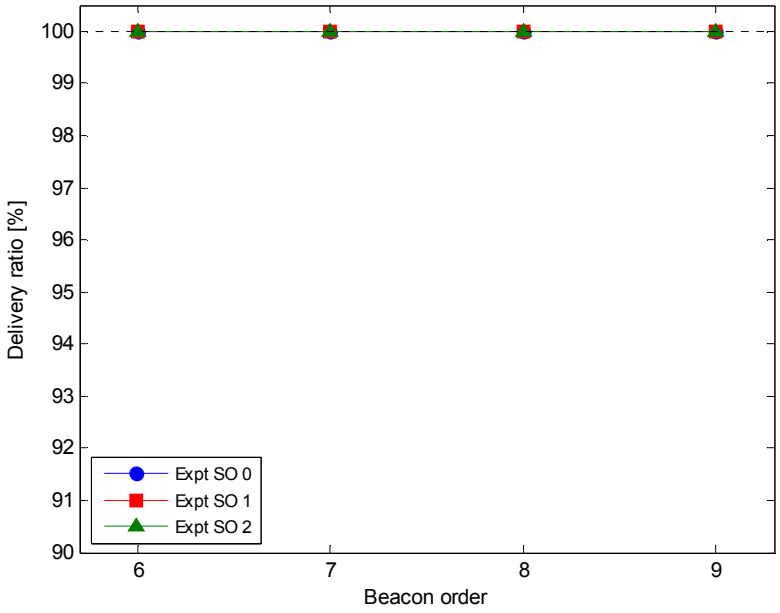


Figure 42 Average delivery ratio of nodes at 60 seconds/packet at superframe orders 0 to 2

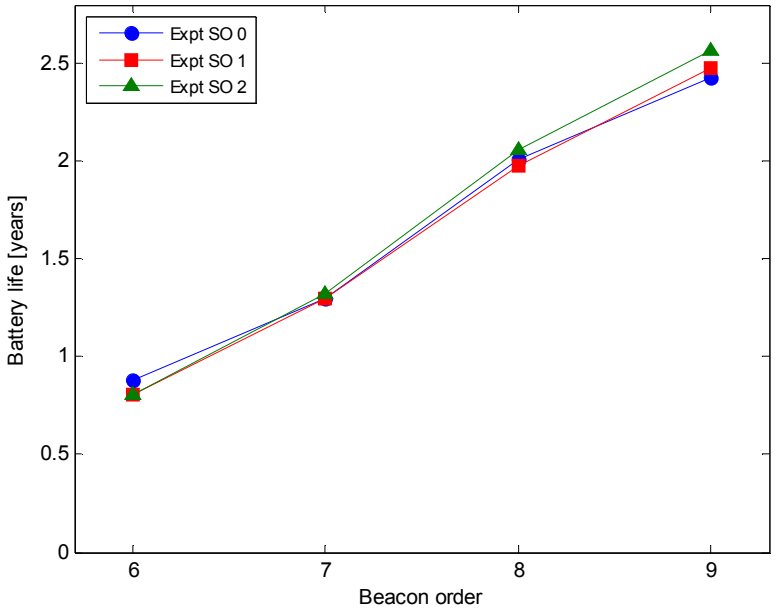


Figure 43 Battery life of node with increasing beacon and superframe orders at data rate 60 seconds/packet

Results at 100 seconds per packet

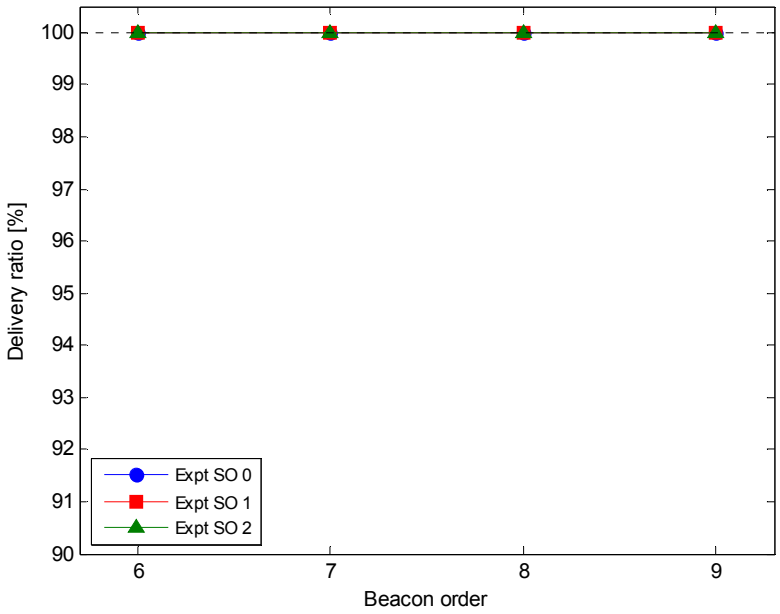


Figure 44 Average delivery ratio of nodes at 100 seconds/packet at superframe orders 0 to 2

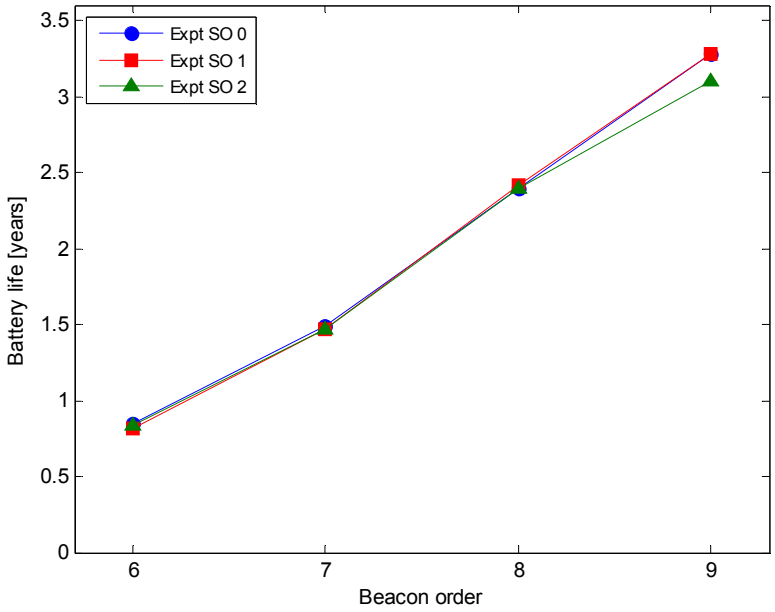


Figure 45 Battery life of node with increasing beacon and superframe orders at data rate 100 seconds/packet

Results at 200 seconds per packet

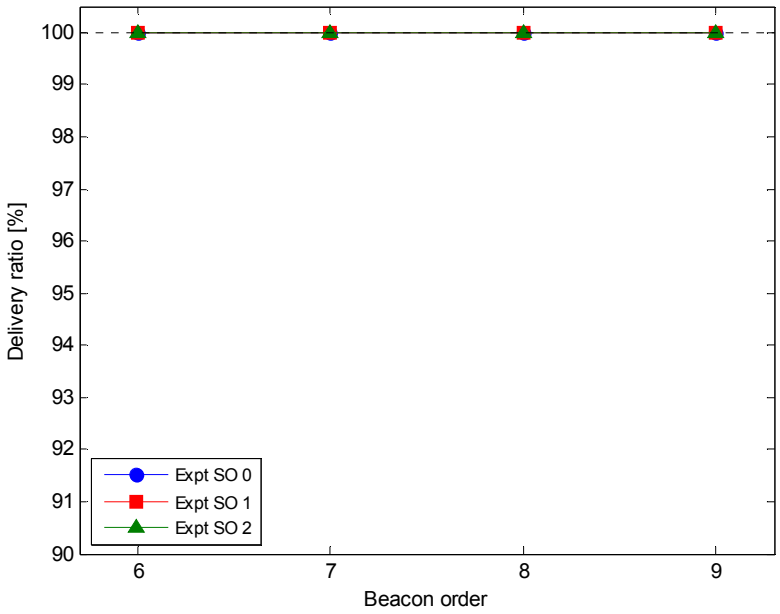


Figure 46 Average delivery ratio of nodes at 200 seconds/packet at superframe orders 0 to 2

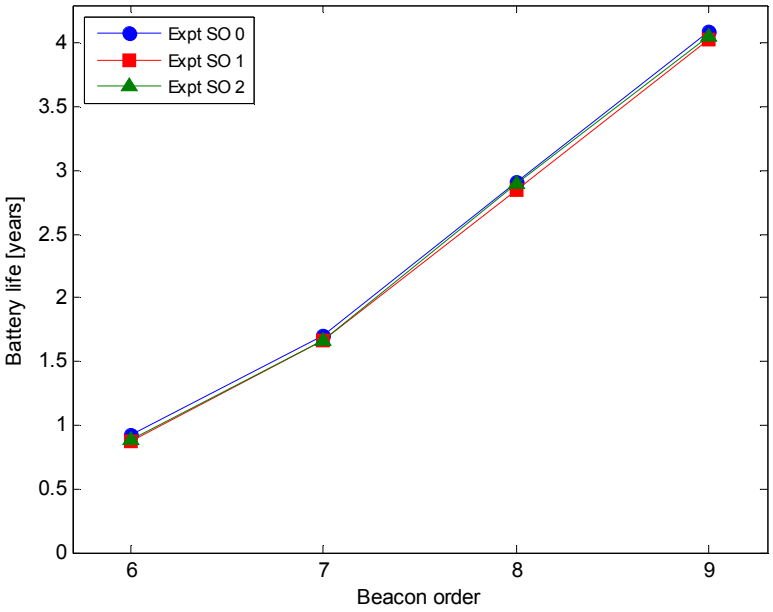


Figure 47 Battery life of node with increasing beacon and superframe orders at data rate 200 seconds/packet

Results at 1000 seconds per packet

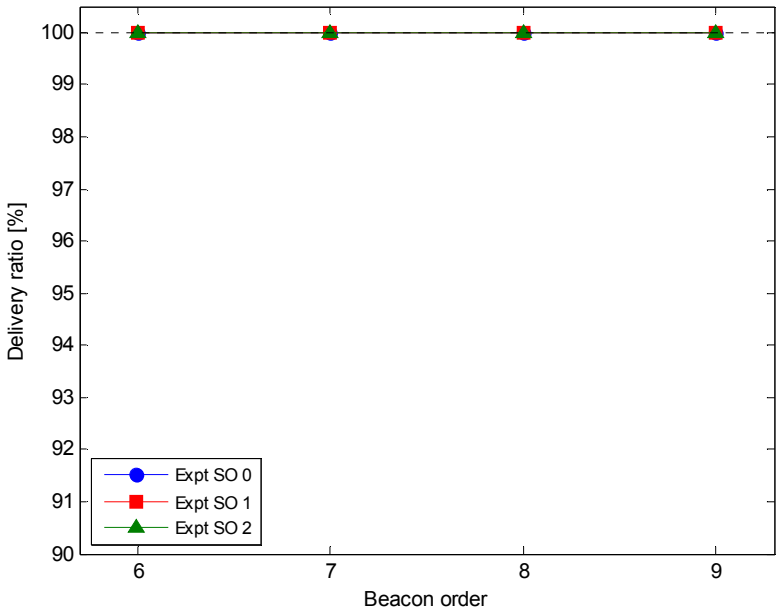


Figure 48 Average delivery ratio of nodes at 1000 seconds/packet at superframe orders 0 to 2

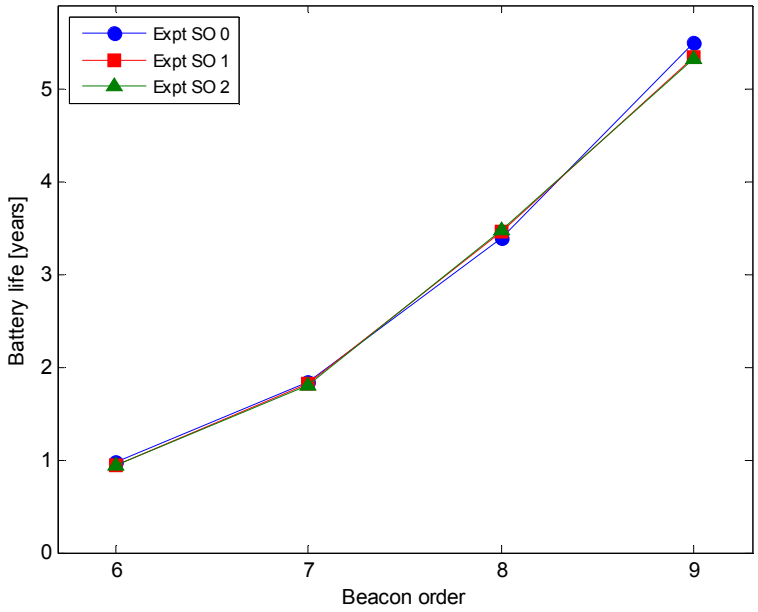


Figure 49 Battery life of a node with increasing beacon and superframe orders at data rate 1000 seconds/packet

4.5.2 Non Beacon Mode

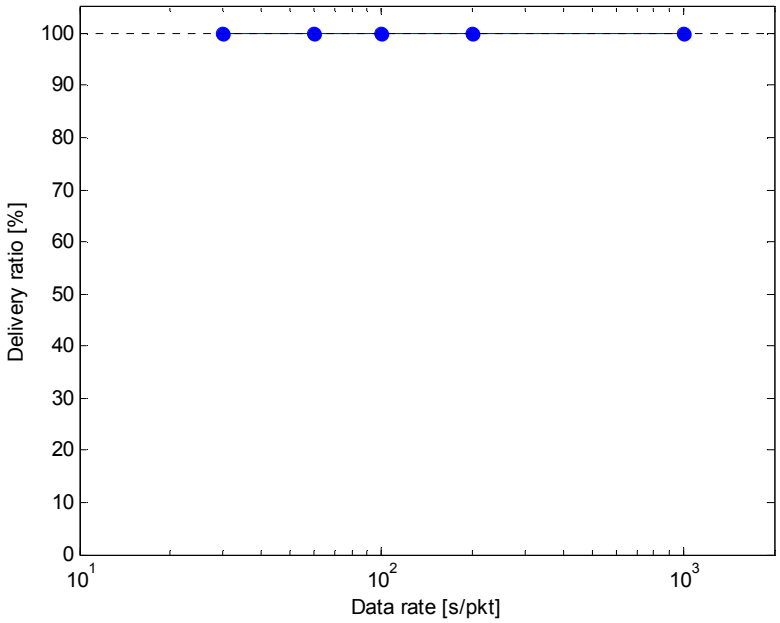


Figure 50 Average delivery ratio of nodes in non beacon mode

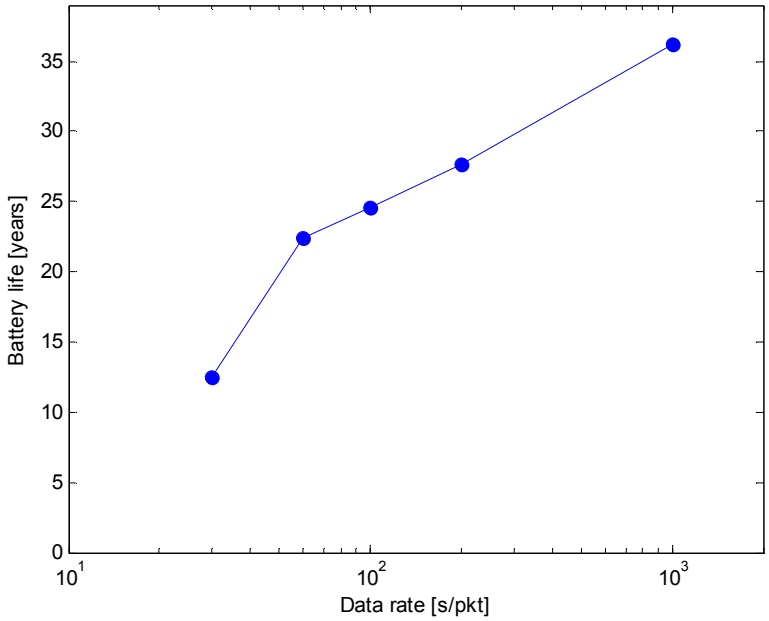


Figure 51 Battery life of a node with decreasing data rate in non beacon mode

4.5.3 Comments on Results

In beacon mode the battery life results increase considerably with beacon order as well as with decreasing data rate. There also minimal difference in battery life with changing superframe orders which is demonstrated in Figure 41, Figure 43, Figure 45, Figure 47, Figure 49, and Figure 51. Non-beacon mode results in a much longer battery life compared to beacon mode which is to be expected since nodes only have to wake up when they intend to transmit data, rather than waking up each beacon interval to listen for a beacon from the coordinator.

Figure 40 shows that 30 seconds/packet was the only data rate to have points where the delivery ratio was not 100%. This figure clearly shows that increasing the superframe order results in an increased delivery ratio. The other data rates of 60, 100, 200 and 1000 seconds/packet exhibit a delivery ratio of 100% for all combinations of beacon order and superframe order as illustrated in Figures Figure 42, Figure 44, Figure 46, Figure 48 and Figure 50.

These initial experimental results were very different from the simulation results. This was a problem as there were many uncertainties as to where the problems could lie. Although many studies had used NS-2 to simulate Zigbee networks, most concentrated on researching different routing aspects and few recorded the energy consumption in their research.

The above series of experimental results show that increasing the superframe order usually only results in a small reduction in battery life; however the results obtained in the second series of simulation results in Section 3.11 show this to be the opposite. This indicates that there are still contention and collision issues in the simulator which increasing the superframe order helps to resolve.

The difference between the battery life results obtained from the simulations and experimental results is difficult to pin point as it could either be a problem with the experiment setup, MATLAB processing script or another issue with the simulator. This is described in the following section.

4.6 Experiment Validation

To verify the accuracy of the acquired data, a sample of the raw data was plotted in MATLAB for better examination. Figure 52 shows a power plot of the end device during a data packet transmission. The time intervals for each mode are described in Table 14 and were obtained from the raw data from the data acquisition module and were validated using the oscilloscope.

While the MATLAB script was being developed, the samples taken were compared to the results of the digital storage oscilloscope; therefore it was known that the data acquisition setup was working correctly.

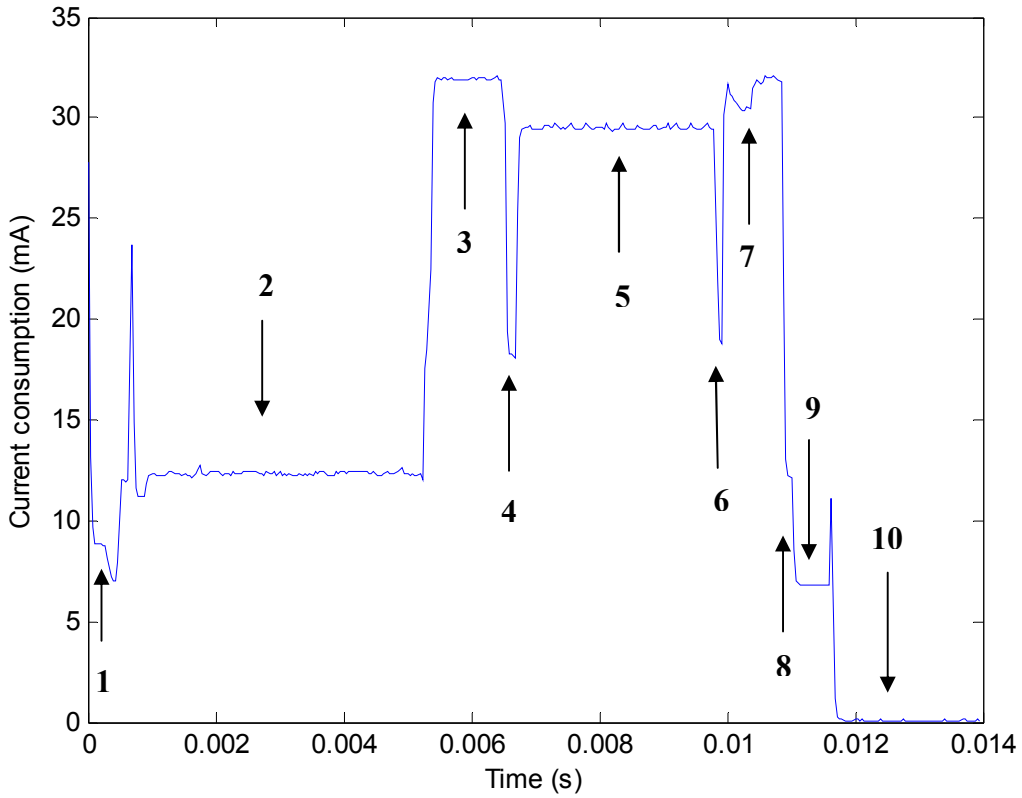


Figure 52 Current consumption of a node during a single packet transmission

Table 14 Current consumption of node during different MAC transceiver states while in active mode

Interval	Description	Current	Duration
1	CPU startup sequence from sleep mode. CPU running on 16MHz clock	8.7mA	0.4ms
2	CPU switched to and running on 32 MHz clock	12.2mA	4.5ms
3	Transceiver on in RX mode, performing CSMA/CA	32mA	1.1ms
4	Switch from RX to TX mode	18.3mA	0.16ms
5	Transmission of data packet, transceiver in TX mode	29.5mA	3.1ms
6	Switch from TX to RX mode	18.9mA	0.8ms
7	Reception of acknowledge packet from Coordinator, transceiver in RX mode	31mA	0.92ms
8	Packet processing, CPU running on 32 MHz clock	12.3mA	0.12ms
9	Shutdown sequence. CPU running on 16 MHz clock	7.0mA	0.6ms
10	Sleep mode. CPU off, sleep timer running from 32 KHz clock.	0.5μA	Until next packet

It was then discovered that the discrepancy between the results was due to the significant CPU processing interval when the microcontroller wakes from sleep mode before the transceiver is enabled. NS-2 is not able to take CPU processing time into account and therefore a significant amount of energy was being accounted for in the data acquisition which was not in the simulation.

In order to overcome this problem, code was added to the MATLAB script which removes the CPU processing time both before and after packet transmission. Upon rerunning the MATLAB script on the previously acquired data, the results were considerably closer to the simulation results and the modification was used for all experiments.

4.7 Comments on Experiments

It became apparent when performing the experiments that the network did not function correctly at beacon order 10. After a period of time, some of the nodes would lose synchronisation with the coordinator and frequently fail to re-associate and subsequently stop sending data. Figure 53 shows what happens to the current consumption of a node which has lost synchronisation with the coordinator. After much debugging the cause of this problem still could not be determined however clock drift is suspected. Because of this problem it was decided to not include beacon order 10 in the above results.

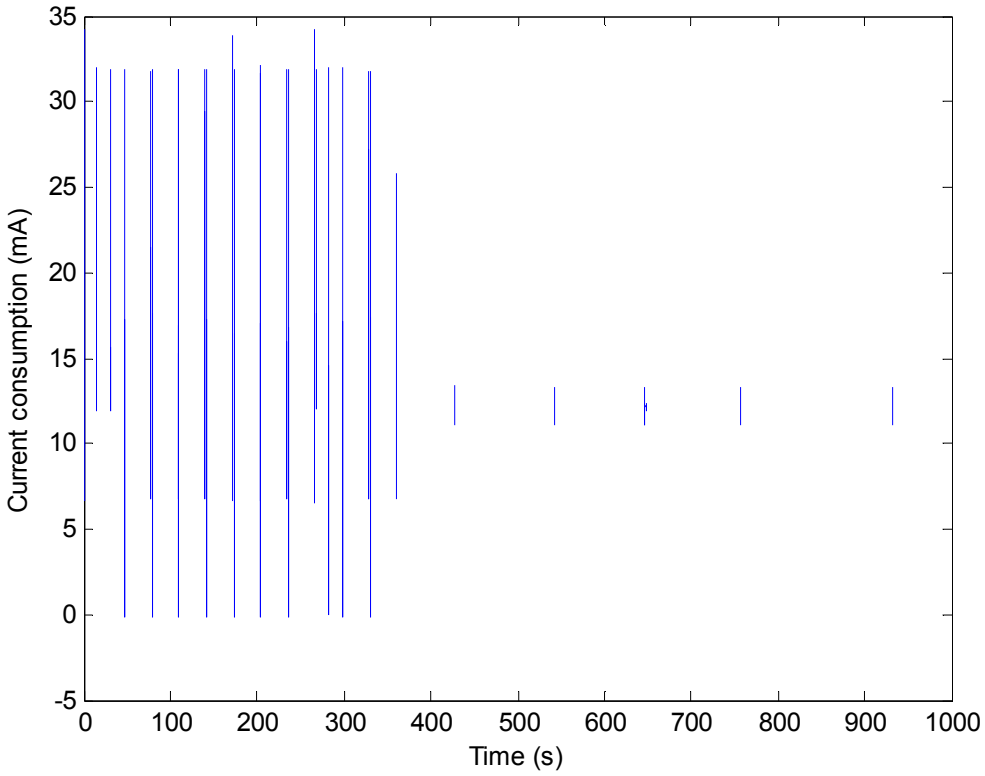


Figure 53 Current consumption of a node at beacon order 10 which shows the device loosing sync with the coordinator at 350 seconds

At beacon orders 9 it became necessary to increase the delay between starting the transmitting nodes from 5 seconds to 15 seconds. This was due to several nodes trying to associate at the same time which resulted in a lot of channel contention. This then meant that a node is not able to associate for several beacon orders and would then stop the association process and thus not send data. Increasing the delay prevented the channel contention and all devices would associate correctly.

It was originally intended to measure the current consumption continuously for the 1000 second duration of the experiment. However as indicated earlier in Section 4.3.7 this generated a huge volume of data which the computer was unable to process due to lack of RAM. Therefore the sampling method described in Section 4.3.7 was implemented. Using this approximation for calculating the sleep mode current consumption could result in a longer than actual battery life due variations in sleep current which the approximation does not take into account.

After changing the sampling method to the above method, the number of samples per packet transmission was set at 1000, which from the results on the oscilloscope appeared to be more than adequate. There turned out to be a special case when the device would wake up, update its internal timer task, then perform the data transmission task. This meant the data acquisition would stop sampling before the transmission had taken place. To solve this problem, the number of samples per packet transmission was increased to 2500 which enabled all the packet transmission data to be captured.

Chapter 5 - Results

5.1 Introduction

This chapter describes the results of this research into IEEE 802.15.4 and Zigbee wireless networks. Both computer modelling and experimental investigation have been performed. Results for the star network topology simulations are validated with results from experimental investigation to both determine optimum network configuration but also validate the accuracy of the simulator. Although the mesh network topology was unable to be simulated due to limitations with the Zigbee NS-2 model, the experiments were repeated in non beacon mode using the mesh and star network topologies on the development boards for further comparison and to further evaluate the usefulness of the technology.

5.2 IEEE 802.15.4 Star Network Topology Results

This section contains the delivery ratio and battery life results for the IEEE 802.15.4 star network topology. Both beacon mode and non beacon mode results are included in this section. Power consumption graphs have not been included in this section as battery life is more human readable but are available in Appendix C.

5.2.1 Beacon Mode

This section contains the delivery ratio and battery life results of the experimental results and simulations in beacon mode at data rates 30, 60, 100, 200 and 1000 seconds/packet which are illustrated in Figure 54 to Figure 69.

Results at 30 seconds per packet

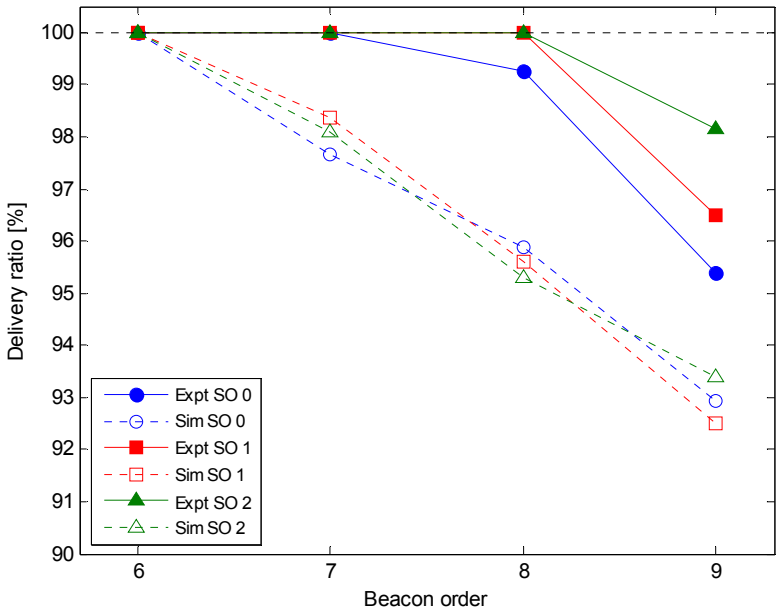


Figure 54 Delivery ratio of experiments and simulations with increasing beacon and superframe orders at data rate of 30 seconds/packet

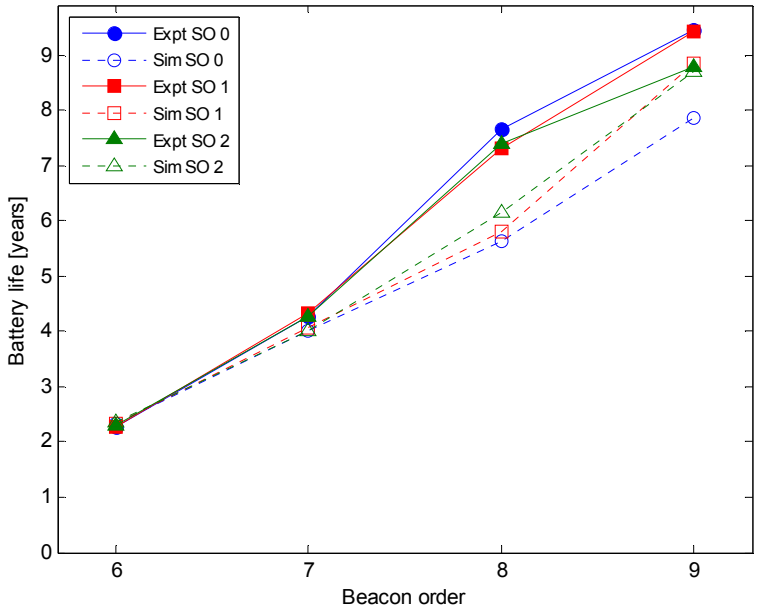


Figure 55 Battery life of experiments and simulations with increasing beacon and superframe orders at data rate of 30 seconds/packet

Results at 60 seconds per packet

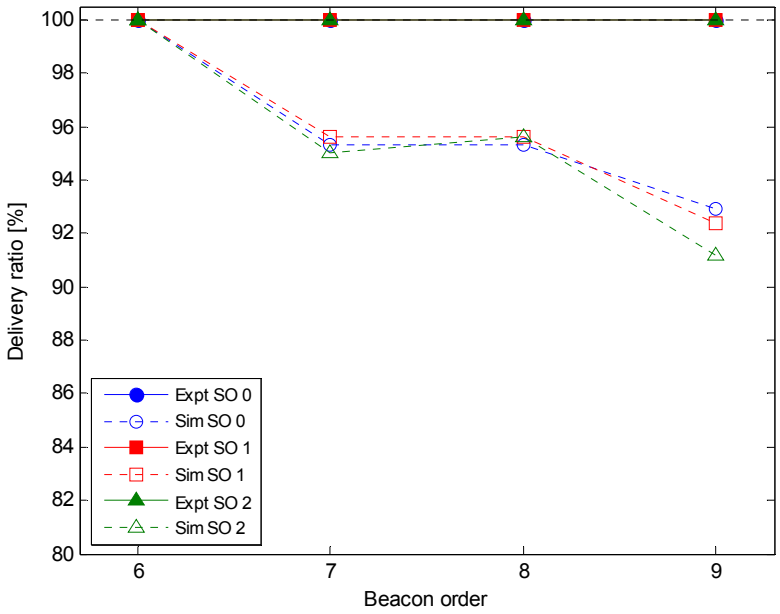


Figure 56 Delivery ratio of experiments and simulations with increasing beacon and superframe orders at data rate of 60 seconds/packet

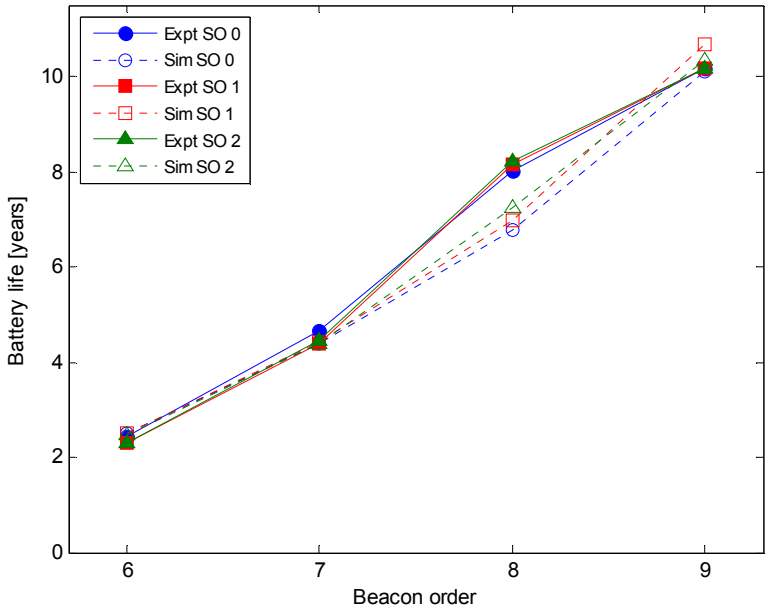


Figure 57 Battery life of experiments and simulations with increasing beacon and superframe orders at data rate of 60 seconds/packet

Results at 100 seconds per packet

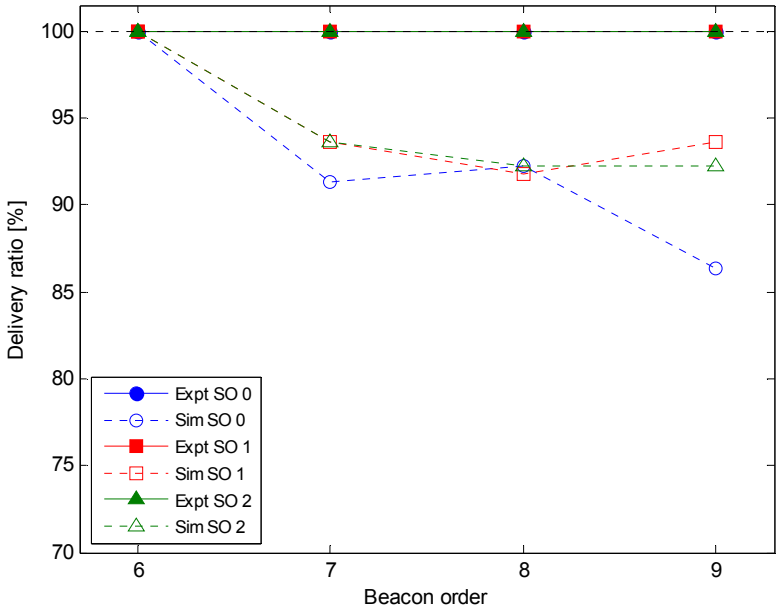


Figure 58 Delivery ratio of experiments and simulations with increasing beacon and superframe orders at data rate of 100 seconds/packet

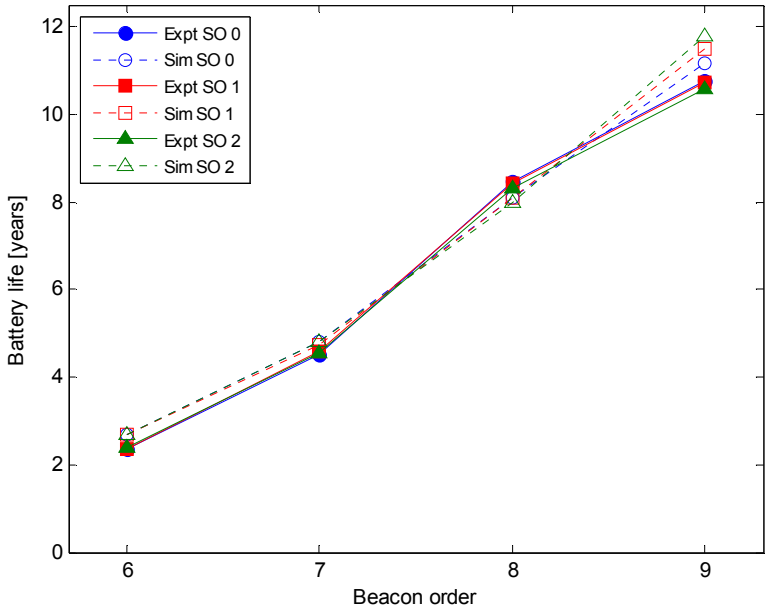


Figure 59 Battery life of experiments and simulations with increasing beacon and superframe orders at data rate of 100 seconds/packet

Results at 200 seconds per packet

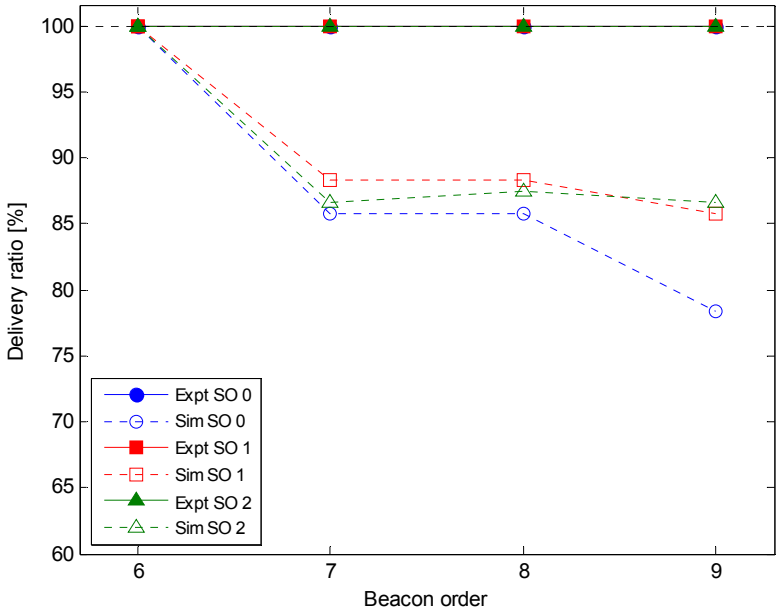


Figure 60 Delivery ratio of experiments and simulations with increasing beacon and superframe orders at data rate of 200 seconds/packet

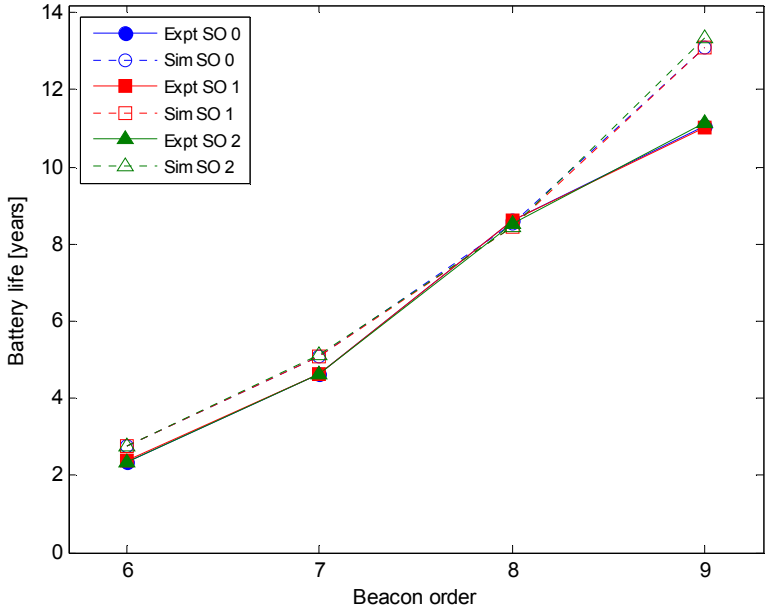


Figure 61 Battery life of experiments and simulations with increasing beacon and superframe orders at data rate of 200 seconds/packet

Results at 1000 seconds per packet

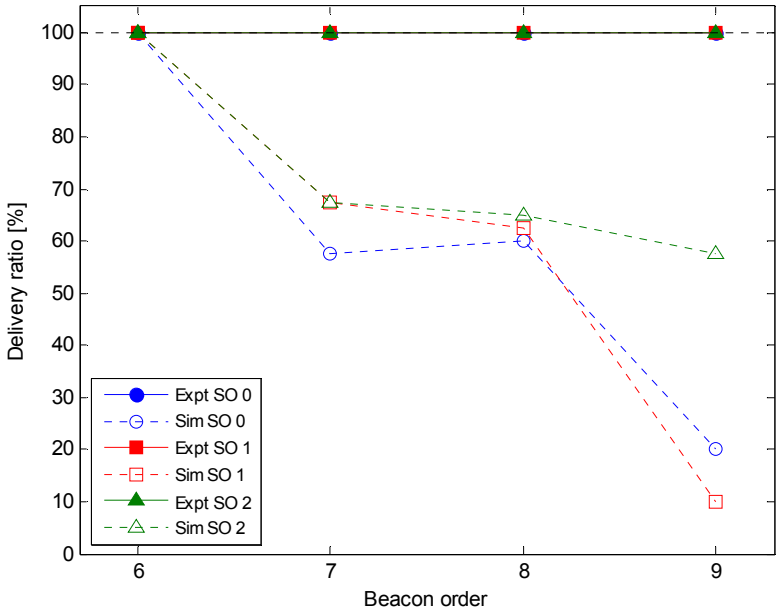


Figure 62 Delivery ratio of experiments and simulations with increasing beacon and superframe orders at data rate of 1000 seconds/packet

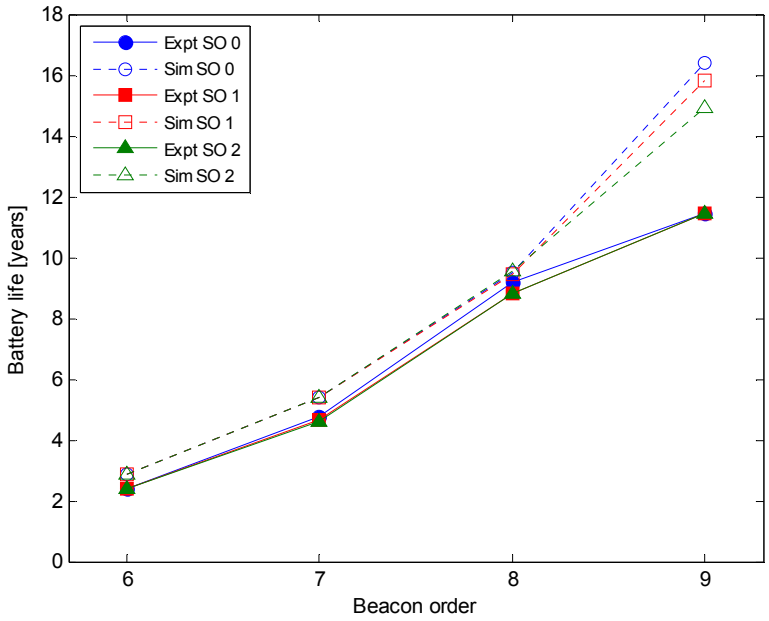


Figure 63 Battery life of experiments and simulations with increasing beacon and superframe orders at data rate of 1000 seconds/packet

While the simulator is often able to estimate the battery life to with 15 percent of the experimental results as shown in Figures 55, 57, 59, 61, 63, the trend is quite different. The simulation results show a consistent increase in battery life with increasing beacon order

while the experimental results show the rate of increase in battery life is initially high but begins to decrease at beacon order 8, showing that the battery life is converging.

The delivery ratio results in Figures 54, 56, 58, 60 and 62 are very different, with the experimental results consistently performing better than the simulations. These results do show that, as expected, increasing the superframe order usually results in an improved delivery ratio with minimal difference in power consumption.

The disparity between the simulation and experimental results is discussed further in Chapter 6.

5.2.2 Non Beacon Mode

This section contains the delivery ratio and battery life results of the experimental results and simulations in non beacon mode at data rates 30, 60, 100, 200 and 1000 seconds/packet which are illustrated in the following two figures.

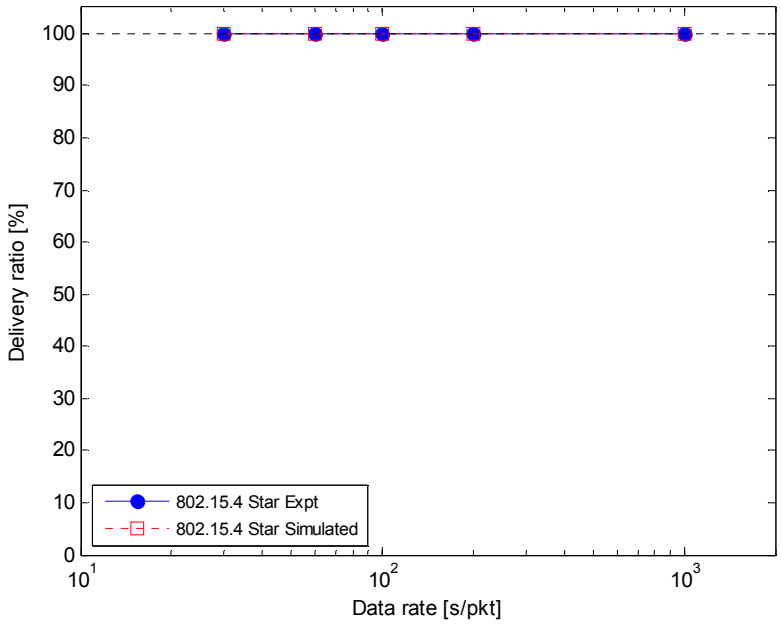


Figure 64 Delivery ratio comparison between simulation and experimental results

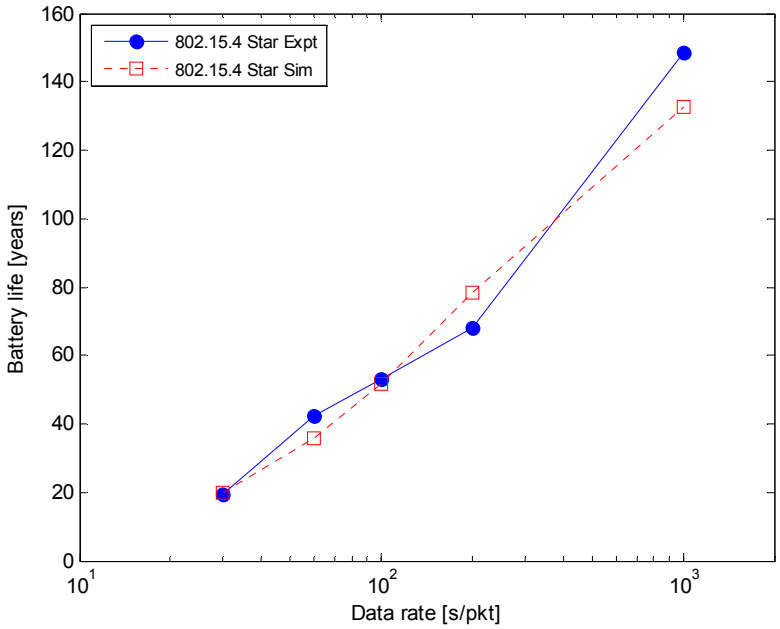


Figure 65 Battery life of simulation and experimental results in non beacon mode

Figure 64 and Figure 65 show that in non beacon mode, the simulator gives much better results compared to simulating beacon mode networks. Delivery ratio was the same as the experimental results with a constant 100% at all data rates.

5.3 Zigbee Network Results

To further analyse Zigbee, the star and mesh topologies were compared. The results shown below in Figure 66 and Figure 67 indicate that at very low data rates, the mesh network topology uses more power due to the additional routing traffic present in the network. However the battery life is still excellent and is longer than the shelf life of the battery at all data rates. Delivery ratio for both networks was identical at 100%.

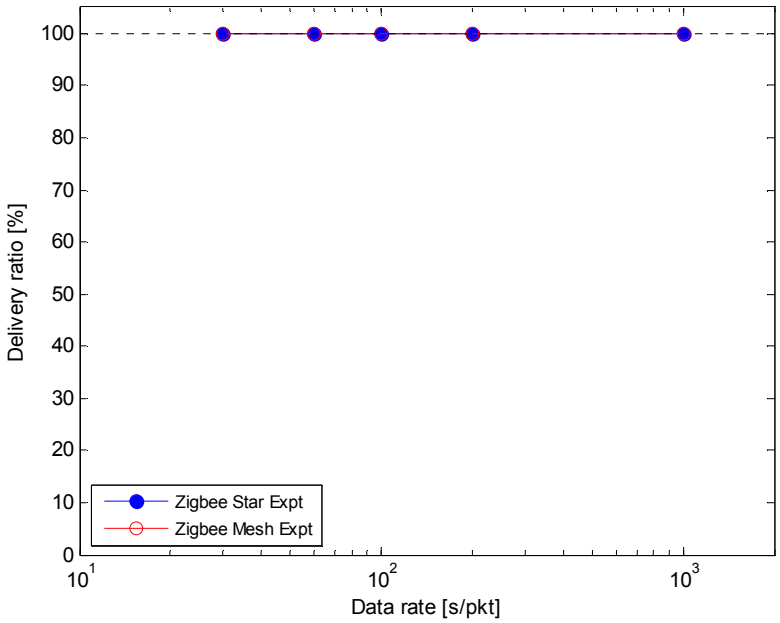


Figure 66 Delivery ratio of Zigbee star and mesh network topologies

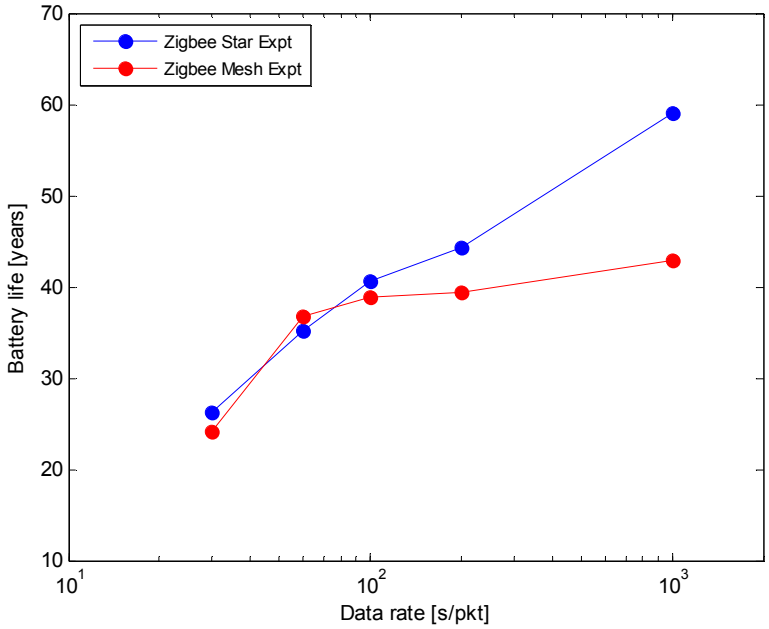


Figure 67 Battery life of Zigbee star and mesh network topologies

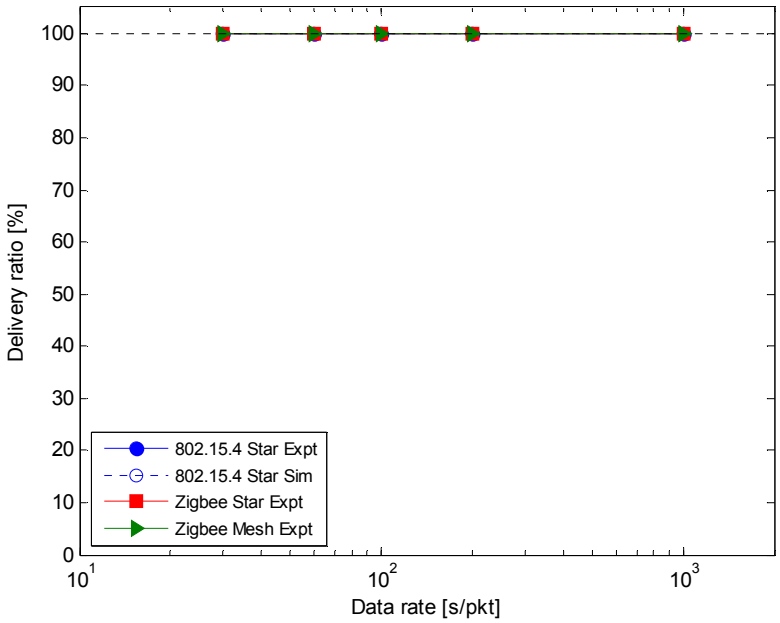


Figure 68 Delivery ratio of 802.15.4 non beacon MAC and Zigbee mesh and star network topologies

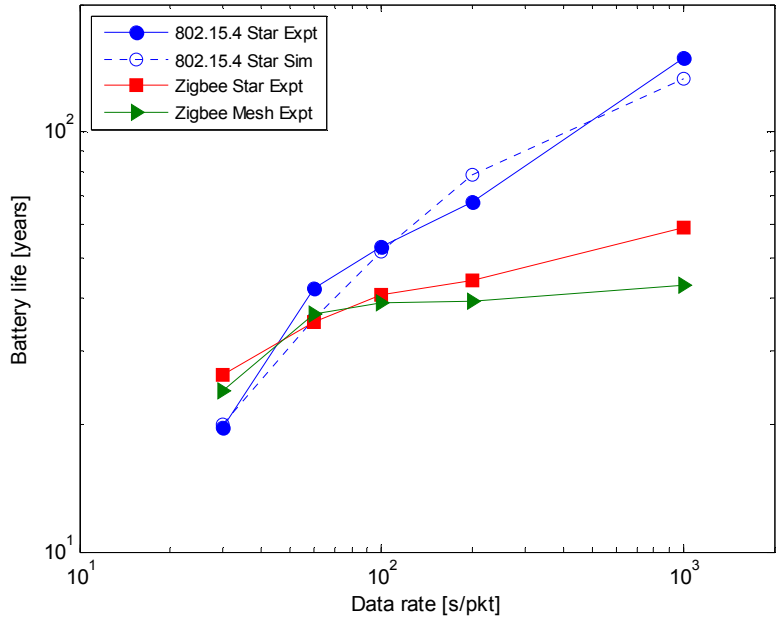


Figure 69 Battery life of 802.15.4 non beacon MAC and Zigbee mesh and star network topologies

As expected, the delivery ratio of the non beacon networks was a constant 100%, as illustrated in Figure 68.

When comparing the three non beacon networks in Figure 69 it can be seen that the IEEE 802.15.4 MAC protocol gives considerably longer battery life overall. The power consumption of all three networks does follow the same trend, with a rapid increase in battery life until the data rate reaches 100 seconds/packet before beginning to level out.

The reason for the shorter battery life in the Zigbee networks versus 802.15.4 MAC is due to the extra routing packets that are broadcast between nodes. When the Zigbee network is configured for the mesh network topology, further routing packets are broadcast which is why the resulting battery life is the lower than the star topology.

5.4 Comments on Results

For beacon mode networks, the modified simulator is able to estimate battery life to within 15 percent, usually with much greater accuracy. However the exception is beacon order 9 at 1000 seconds/packet where the error is around 20 percent.

Although the difference in battery life results was usually within 15 percent, the simulated delivery ratio exhibited a totally different trend compared to the experimental results the trend was quite different compared to the experimental results. The experimental results usually had a 100 percent delivery ratio whereas the simulation results began to decrease as soon as the beacon order increased. It can be seen that the simulation results in this chapter have improved considerably from the original simulation results after modifying the simulator as described in Section 3.10.

As noticed in the earlier results, when operating in beacon mode, particular data rates and beacon orders in both the simulation and experimental results, increasing the superframe order gives better battery life (ie lower power consumption) than at a lower superframe order due to the reduced channel contention and reducing retransmissions as well as improving the delivery ratio. This was also discovered by Kohvakka *et al* [6].

In non beacon mode, the simulator produces good power consumption results which closely follow the experimental results and the delivery ratio is exactly the same as the experimental results.

Interestingly, the experiments frequently outperform the simulated results, particularly in beacon mode, for delivery ratio and battery life. This is usually not the case and is especially surprising since the experiments took place in a real industrial environment with metal structures present such as tables and shelves and no direct line of sight path. It is suspected that there is a problem with the event scheduler in NS-2 which is causing the poor delivery ratio results. This is because when the network is operating in beacon mode there are additional events relating to the transmission and reception of beacons as well as events for transmitting data and operating in beacon mode gives delivery ratio results below the experiments. Debugging the NS-2 trace file indicates, although it is not entirely clear, that the scheduler is causing the packet collisions. Unlike in beacon mode when in non-beacon mode, no packets are lost by the simulator. Clearly the model that the simulator is using needs further work to enable more realistic simulations, especially when used in beacon mode networks.

Although this research focuses on a fixed 64 byte packet size, adjusting this size will also change the power consumption (and subsequently battery life) due to the change in the transmission length with a longer packet resulting in higher power consumption, this has been demonstrated by other researchers.

The number of network nodes will also have an affect on the power consumption. Previous research has shown that increasing the number of nodes also increases the average power consumption of each node due to extra network contention and collisions for both beacon and non beacon networks.

Chapter 6 - Discussion and Conclusion

6.1 Introduction

This chapter discusses the results of the computer simulations and the experimental results for the power consumption, battery life and delivery ratio of IEEE 802.15.4/Zigbee wireless networks.

6.2 Problems

Many studies that have been undertaken so far analyse beacon mode operation of Zigbee using NS-2, however few have compared the results with a real beacon mode network. While attempting to create a beacon enabled Zigbee network using the Chipcon CC2430BB development boards, Z-stack would refuse to load on the coordinator. Debugging the source code was not helped by the fact that the function causing the problem was hidden in a compiled library and thus not able to be accessed.

After asking Texas Instruments for advice, it turns out that although provision has been made to enable beacons and the data sheet for the stack gives details on how to change the beacon order, the stack does not actually support beacon mode.

This problem was not anticipated as the decision to use these devices was primarily based on the documentation showing how to change the beacon order, thus it was assumed it was possible. Since this is a key component of my research an alternative was required. Since beacons cannot be used in a mesh network it was decided to use the Texas Instruments MAC (TiMAC) software to enable the star network topology to be tested. Beacons in TiMAC did work correctly and enabled beacon mode with star network topology to be tested.

It was soon discovered that the Zigbee/IEEE 802.15.4 model in NS-2 did not fully support Zigbee routing or mesh network topology. This could be why the majority of studies about Zigbee that use NS-2 are only using the star network topology. Since the aim of this research

was not writing NS-2 simulation models it was decided to simply simulate the scenarios that were possible.

During examination of the NS-2 network trace file it was discovered that there was an excessive number of AODV routing and ARP packets being transmitted between nodes and the coordinator. Preliminary hardware experiments on the Zigbee network while using a packet sniffer showed no such AODV or ARP packet activity after device association. Whenever a node transmits or receives packets in NS-2 it decrements the energy variable of the nodes involved. Because of this, these extra packets were consuming a significant amount of energy and thus the simulations were giving very poor battery life.

It was noticed while working with the simulator, adding more than a few nodes caused a large drop in the delivery ratio of packet transmissions when using non beacon mode and to a lesser extent in beacon mode. After much investigation and trace file analysis it appears to be a problem with the event scheduler in NS-2 as nodes appear to try to send packets at the same time as each other, even when they are configured to send at different time intervals and start at different times.

As evident in the results in the previous section, there is still collision problems with the simulator when operating in beacon mode. As described earlier it is suspected that this is due to issues with the NS-2 event scheduler however this has not been proven and needs further investigation.

Another problem that was encountered while developing the firmware for the Zigbee network was that the end devices were not entering sleep mode correctly and subsequently using excessive amounts of power. Z-stack uses a real time operating system and power management is a key component of the operating system. When no tasks are scheduled the operating system calls a sleep task which powers off the peripherals and puts the microcontroller to sleep for a predetermined amount of time. When the CC2430 is put into sleep mode, certain address ranges in its RAM do not retain data. To prevent RAM data being lost, the sleep task checks to see if the stack pointer has exceeded the boundary into the volatile part of RAM and if it has it prevents sleep mode being entered. Due to a bug in the source code, sleep mode was only sometimes being entered and once the change was made the device was sleeping correctly.

6.3 Conclusions

This research has clearly shown that in single sink IEEE 802.15.4/Zigbee wireless networks, non beacon mode gives the longest battery life (lowest power consumption) and the best delivery ratio at all tested data rates in both computer simulations and the experimental investigation.

The main contributions of this research to the field of IEEE 802.15.4/Zigbee wireless networks are:

- A comparison between beacon and non beacon operation of the IEEE 802.15.4 MAC layers.
- Comparison between hardware experiments and results generated by the popular computer network simulation tool NS-2.
- Comparison between Zigbee mesh and star network topologies
- Comparison between IEEE 802.15.4 MAC and Zigbee network stacks in terms of power consumption and delivery ratio.

From the results presented in this thesis, IEEE 802.15.4 in beacon mode results in significantly lower battery life (higher power consumption) and a poorer delivery ratio. However it is likely that in applications where there is bidirectional traffic, the synchronisation provided in beacon mode could be an advantage and could result in fewer collisions than in non beacon mode.

Interestingly, the initial computer simulation results were vastly different from the experimental results. This was a surprise as this simulator had been used for many Zigbee research projects. In order to get reasonable results from NS-2, modifications were required to the AODV and ARP models as well as the energy model in the wireless node class. After performing the modifications described in this thesis, good results were obtained from the simulator which are usually within 15 percent of the experimental results.

However, even after modifying the simulator, the final results show that frequently the experiments out perform the simulation results, even though the experiments were conducted in an industrial environment. Normally simulated results significantly out perform experiments. As mentioned earlier, the IEEE 802.15.4 model in NS-2 needs further work to improve its performance and this could be the subject of future work.

As shown by other researchers [3, 6], it is demonstrated that when operating the network in beacon mode, it is very important to set the beacon order and superframe order correctly, otherwise excessive power consumption and or poor delivery ratio will result. In order to set these parameters correctly, an idea of the expected traffic rate is required prior to configuring the network.

When using the fully featured Zigbee network stack it can be seen the network topology does affect the power consumption, thus battery life of the end devices, with the mesh network topology having a lower battery life than an equivalently sized star network. Even when using the mesh network topology the resulting battery life is likely to be greater than the shelf life of the battery. Therefore the advantages offered by the mesh network topology like reduced congestion and network redundancy can be utilised without compromising the obtainable battery life.

It was also shown that the extra network traffic that is generated when using the fully featured Zigbee stack versus the IEEE 802.15.4 MAC layer software does reduce battery life considerably at very low data rates, however the reduction in battery life is irrelevant as the average shelf life of a battery is considerably less than the calculated battery life.

It was interesting that the initial results were so poor considering that NS-2 has been used in several research papers which show good results, but do not make any mention of initial problems with the simulator. It is suspected the reason that other published work shows good results because the majority study beacon mode which, initially, gave considerably better results than non beacon mode. Typically their research is concentrated at lower beacon orders and higher superframe orders. This increases the duty cycle of the nodes which improves throughput and delivery ratio considerably. However having a high superframe order (thus a high duty cycle) defeats the aim of a low power technology as the power consumption of the devices becomes considerable.

This research clearly demonstrates that the power consumption of the transceiver is almost negligible as it will typically outlast the shelf life of a battery several times. The battery life of a device will be limited by other associated electronics, not by the transceiver itself.

Overall, this research shows that Zigbee and its underlying IEEE standard 802.15.4 is an excellent technology which can achieve superb battery life and delivery ratio results, and is ideal for low data rate battery powered wireless networks and will work very well in the intended single sink real time monitoring application.

6.4 Future Work

While this thesis has been successful in determining optimum network configuration with lowest power consumption in IEEE 802.15.4/Zigbee wireless networks, there are a number of developments that could build on this research.

The IEEE 802.15.4 simulation model for NS-2 could be further modified to improve power consumption and delivery ratio results.

A model of the Zigbee network stack could be developed for NS-2 which would allow the simulation of Zigbee mesh networks. This would be extremely helpful since the mesh network architecture is the most useful and widely used.

Further experiments could be undertaken with more nodes and a greater network depth to investigate the effect on power consumption and delivery ratio.

The experiments and scenarios presented in this research could be repeated with bidirectional traffic, for example, traffic from both the end devices to the coordinator and the coordinator to the end devices.

For increased accuracy, a more powerful computer with more RAM could be employed to allow the data acquisition module to sample continuously for the duration of the experiment. With a more powerful computer, accuracy could be increased further by measuring power consumption of several nodes at once and averaging the results to get a better idea of the average power consumption at a particular operating condition.

Chapter 7 - References

1. Jianliang Zheng, Myung J. Lee. "Will IEEE 802.15.4 Make Ubiquitous Networking a Reality?" *IEEE Communications Magazine*, p. 140-146, June 2004.

2. "The Network Simulator - NS-2 ".v2.33 Available from: http://www.isi.edu/nsnam/ns/ [cited 16 Feburary 2008]

3. Huang Yu-Kai, Pang Ai-Chun. "A Comprehensive Study of Low-Power Operation in IEEE 802.15.4". *MSWiM '07: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, ACM, p. 405-408, 2007.

4. Wu Ling-xi, Zhan Jie, Shi Wei. "The Low Power Study Based on Zigbee Network". *WiCOM 2008: 4th International Conference on Wireless Communications, Networking and Mobile Computing*, IEEE, p. 1-4, 2008.

5. Zhijia Chen, Chuang Lin, Hao Wen, Hao Yin. "An Analytical Model for Evaluating IEEE 802.15.4 CSMA/CA Protocol in Low-Rate Wireless Application". *21st International Conference on Advanced Information Networking and Applications Workshops*, IEEE, p. 899 - 904, 2007.

6. Mikko Kohvakka, Mauri Kuorilehto, Marko Hannikainen, Timo D. Hamalainen. "Performance analysis of IEEE 802.15.4 and Zigbee for large scale wireless network sensor applications". *PE-WASUN 2006: Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, ACM, 2006.

7. Joe Hoffat, Kevin Klues, Obi Orjih. "Configuring the IEEE 802.15.4 MAC layer for single sink wireless sensor network applications". Washington University, St. Louis, Missouri, 2005. Available from: http://www.cs.wustl.edu/~joeh/802_15_4_Eval_Report.pdf [cited 20 March 2008]

8. Bruno Bougard, Francky Catthoor, Denis C. Daly, Anantha Chandrakasan, Wim Dehaene. "Energy Efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: Modelling and improvement perspectives". *Design, Automation and Test Conference in Europe*, p. 196-201, 2005.

9. Zigbee Alliance. "Zigbee Specification". 2006. Available from: http://www.zigbee.org/en/spec_download/download_request.asp [cited 1 August 2008]

10. Jin-Shyan Lee. "An Experiment on Performance Study of IEEE 802.15.4 Wireless Networks". *ETFA 2005: 10th IEEE Conference on Emerging Technologies and Factory Automation*, p. 451-458, 2005.

11. T. Ryan Burchfield, S. Venkatesan, Douglas Weiner. "Maximizing Throughput in Zigbee Wireless Networks through Analysis, Simulations and Implementations".

University of Texas at Dallas, 2007. Available from: www.utd.edu/~rxb023100/pubs/ZigBee_Throughput.pdf [cited 30 November 2008]

12. Chandramani Kishore Singh, Anurag Kumar, P.M Ameer. "Performance evaluation of an IEEE 802.15.4 sensor network with a star topology". *Wireless Networks*. 2007, vol. 14, no. 4, p. 543-568.

13. Vaddina Prakash Rao. "The simulative Investigation of Zigbee/IEEE 802.15.4", Dresden University of Technology, 2005.

14. S.A. Khan, H. Aziz, S. Maqsood, S. Faisal. "Clustered home area network: A beacon enabled IEEE 802.15.4 approach". *ICET 2008: Emerging Technologies*, p. 193 -198, 2008.

15. Miaoqi Fang, Jian Wan, Xianghua Xu, Guangrong Wu. " System for temperature monitor in substation with ZigBee connectivity". *11th IEEE International Conference on Communication Technology*, p. 25-28, 2008.

16. IEEE Computer Society. "802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)". 2006. Available from: [cited

17. Sinem Coleri Ergen. "Zigbee/IEEE 802.15.4 Summary". Advanced Technology Lab of National Semiconductor, 2004. Available from: http://www.sinemergen.com/zigbee.pdf [cited 1 Feburary 2009]

18. "OPNET Modeler". Available from: http://www.opnet.com/solutions/network_rd/modeler.html [cited 30 April 2008]

19. "OMNet++". Available from: http://www.omnetpp.org [cited 5 July 2008]

20. Feng Chen, Falko Dressler. "IEEE 802.15.4 Model for OMNet++/INET Framework". 2008. Available from: http://www7.informatik.uni-erlangen.de/~fengchen/omnet/802154/index.shtml [cited 12 July 2008]

21. Jianliang Zheng, Myung J. Lee. "NS-2 simulator for 802.15.4".v1.1, The City University of New York, Available from: http://ees2cy.engr.cnny.cnny.edu/zheng/pub/file/wpan11.tar.gz [cited 18 Feburary 2008]

22. Iyappan Ramachandran. "Changes made to the IEEE 802.15.4 NS-2 Implementation". University of Washington, 2006. Available from: http://www.ee.washington.edu/research/funlab/802_15_4/ [cited 1 September 2008]

23. Teerawat Issariyakul, Ekram Hossain. *Introduction to Network Simulator NS-2*, Springer, 2008.

24. Kevin Fall, Kannan Varadhan. "The VINT Project - The *ns* Manual". Available from: http://www.isi.edu/nsnam/ns/doc/index.html [cited 1 June 2008]

25. Antenova. "Titanis 2.4 GHz Swivel SMA Antenna". 2008. Available from: http://www.antenova.com/?id=753 [cited 17 August 2008]

26. Theodore S. Rappaport. *Wireless Communications Principles & Practice*. First Edition, Prentice Hall, 1996.

27. "The *ns* Manual - Some typical values of path loss exponent β - Table 18.1". 2009. Available from: http://isi.edu/nsnam/ns/doc/node220.html [cited 17 August]

28. "The *ns* Manual - Some typical values of shadowing deviation - Table 18.2". 2008. Available from: http://www.isi.edu/nsnam/ns/doc/node220.html [cited 17 August]

29. "gnuplot".v4.2.4 2008. Available from: www.gnuplot.info [cited 2 June 2008]

30. Jianliang Zheng. "Low Rate Wireless Personal Area Networks (LR-WPANs) FAQ". The City University of New York, 2006. Available from: http://ee.ccny.cuny.edu/zheng/pub/Pages/wpan-ns2-faq.htm [cited 1 September 2008]

31. Texas Instruments. "CC2430 Datasheet, A True System-on-Chip solution for 2.4GHz IEEE 802.15.4/Zigbee". 2008. Available from: http://focus.ti.com/docs/prod/folders/print/cc2430.html [cited 3 September 2008]

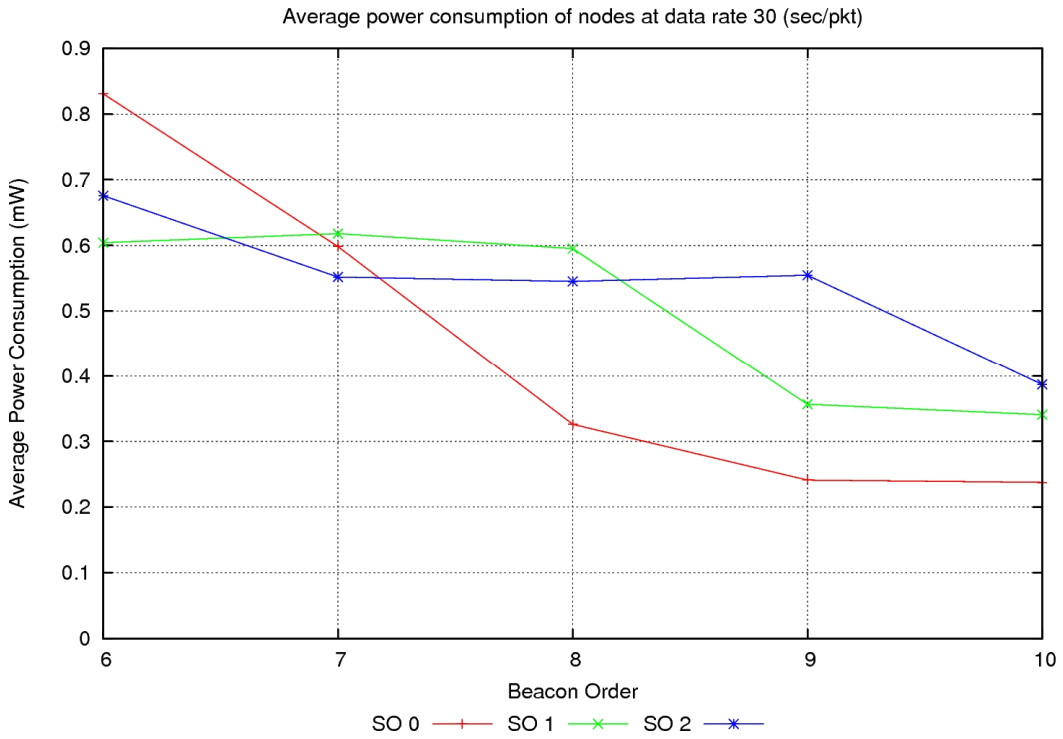
32. Texas Instruments. "CC2431DK - CC2431 Development Kit". Available from: http://focus.ti.com/docs/toolsw/folders/print/cc2431dk.html [cited 20 Feburary 2009]

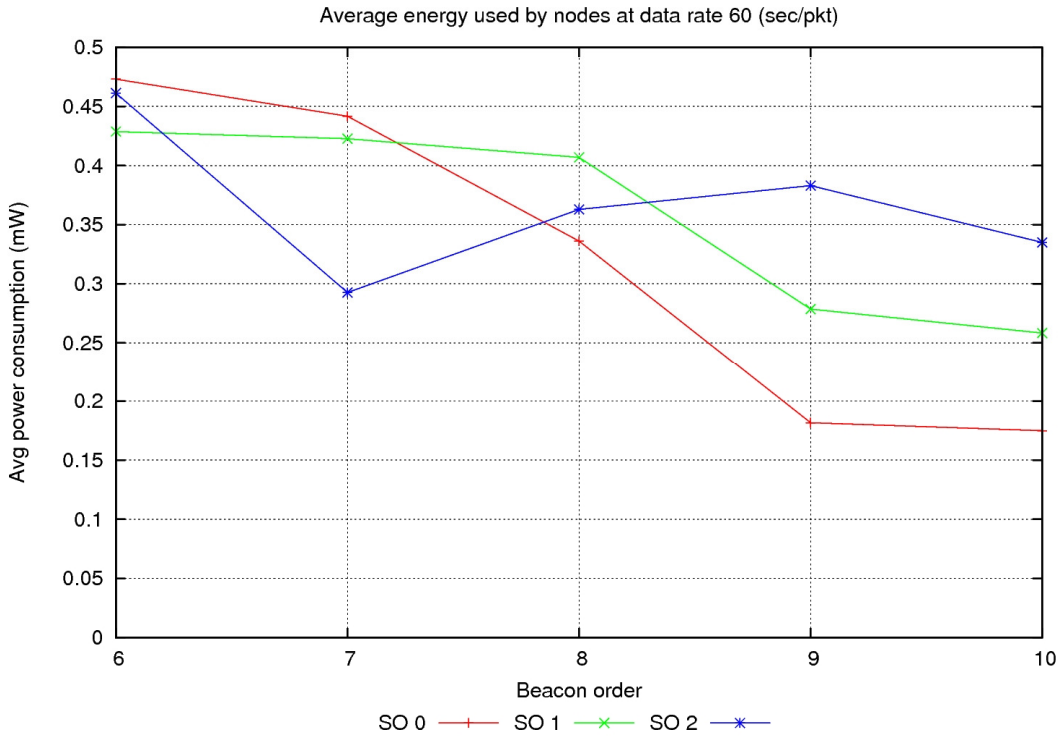
33. "Texas Instruments Z-Stack".v1.4.3 2008. Available from: http://focus.ti.com/docs/toolsw/folders/print/z-stack.html [cited 1 September 2008]

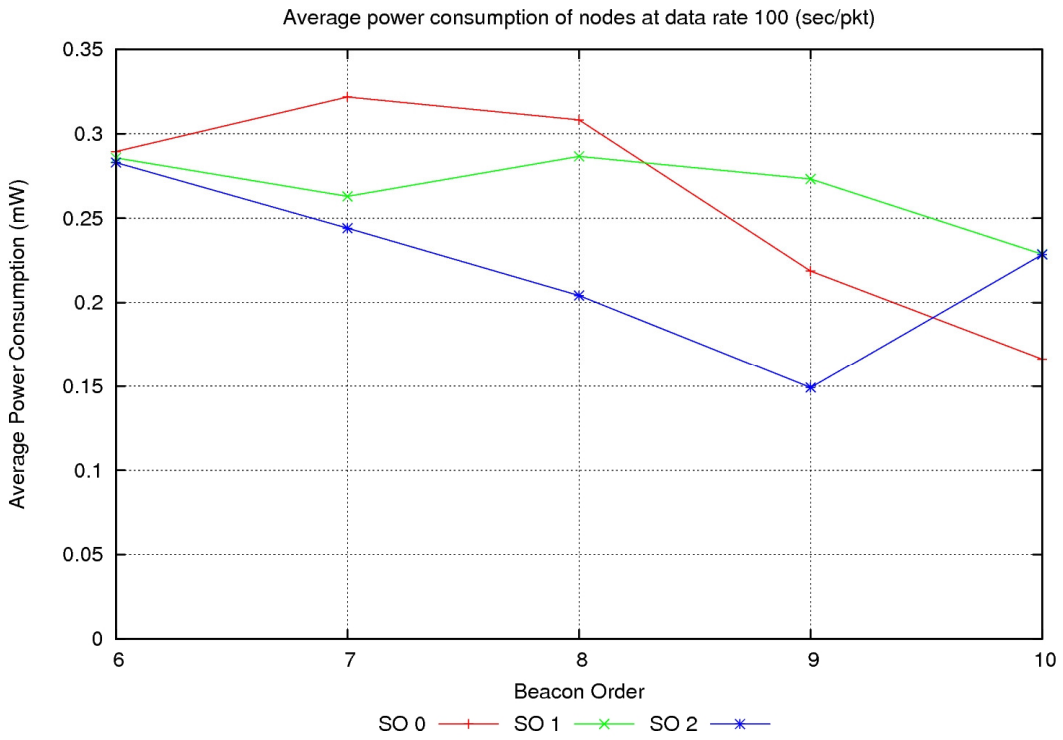
34. "Texas Instruments TiMAC".v1.2.1 2008. Available from: http://focus.ti.com/docs/toolsw/folders/print/timac.html [cited 10 October 2008]

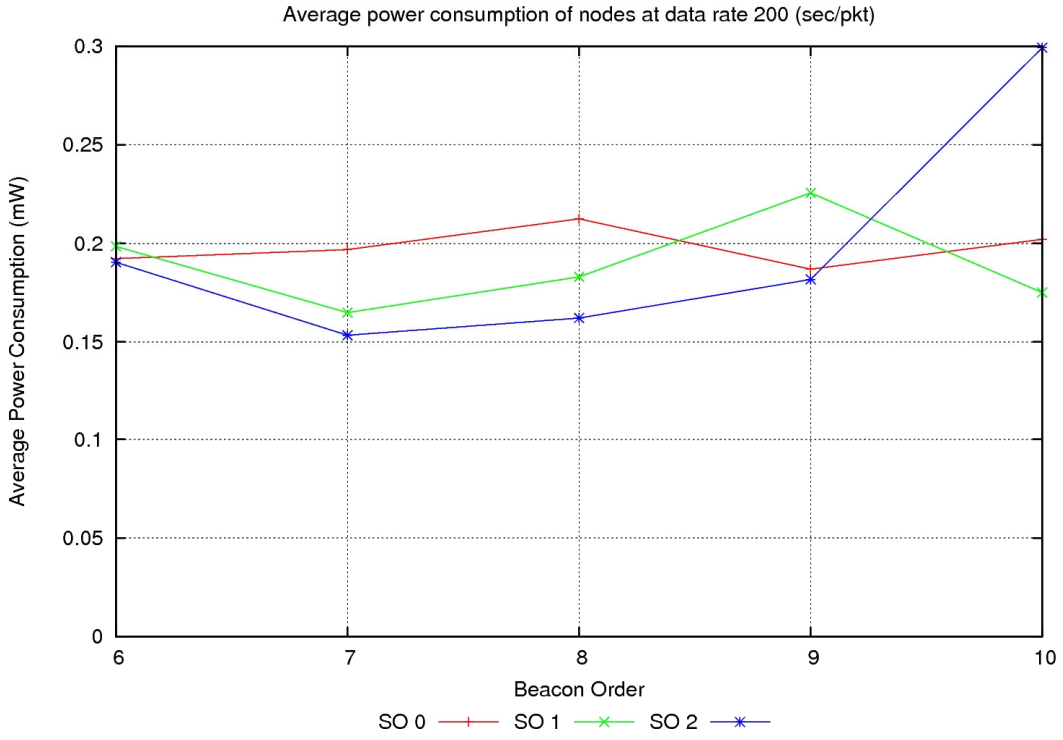
Appendix A - Initial Power Consumption Results

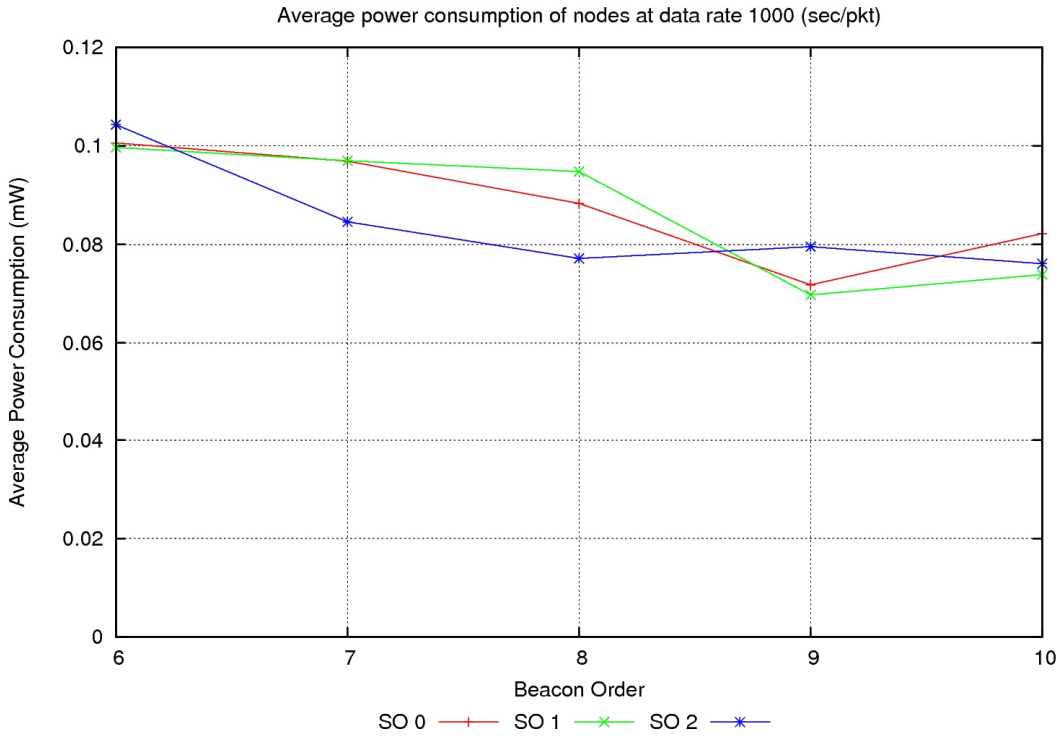
The following graphs are the average power consumption of nodes in beacon mode



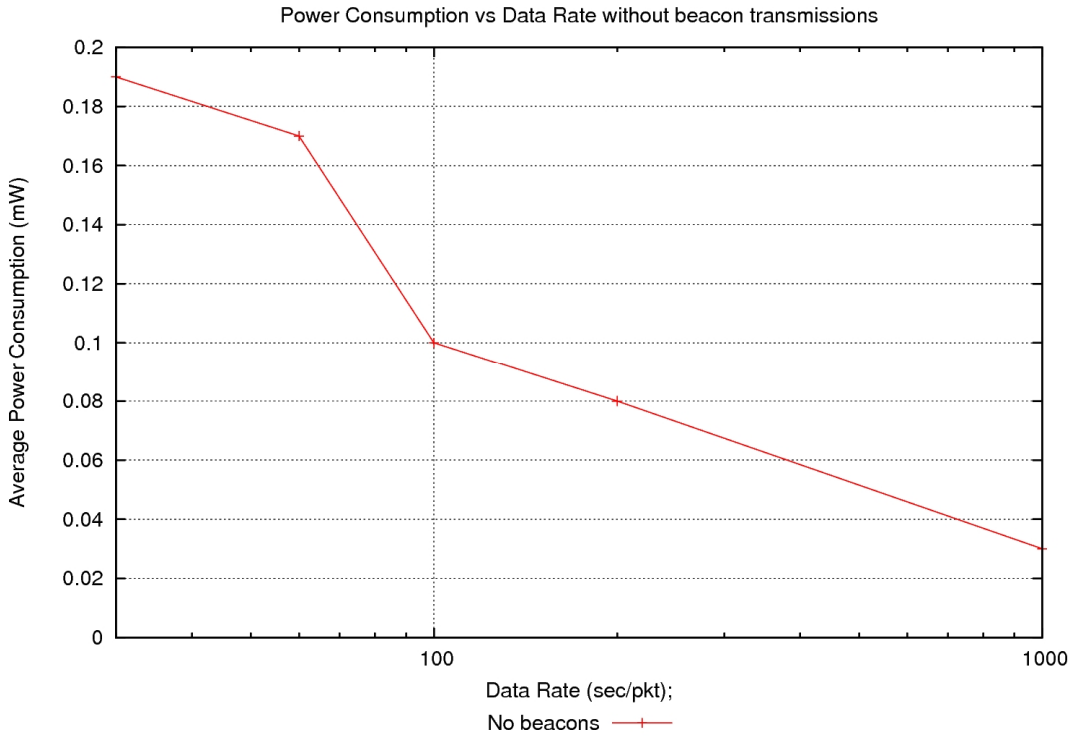



Average power consumption of nodes at data rate 100 (sec/pkt)
Average Power Consumption (mW)
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
6
7
8
9
10
Beacon Order
SO 0
SO 1
SO 2


Average power consumption of nodes at data rate 200 (sec/pkt)
Average Power Consumption (mW)
0.3
0.25
0.2
0.15
0.1
0.05
0
6
7
8
9
10
Beacon Order
SO 0
SO 1
SO 2

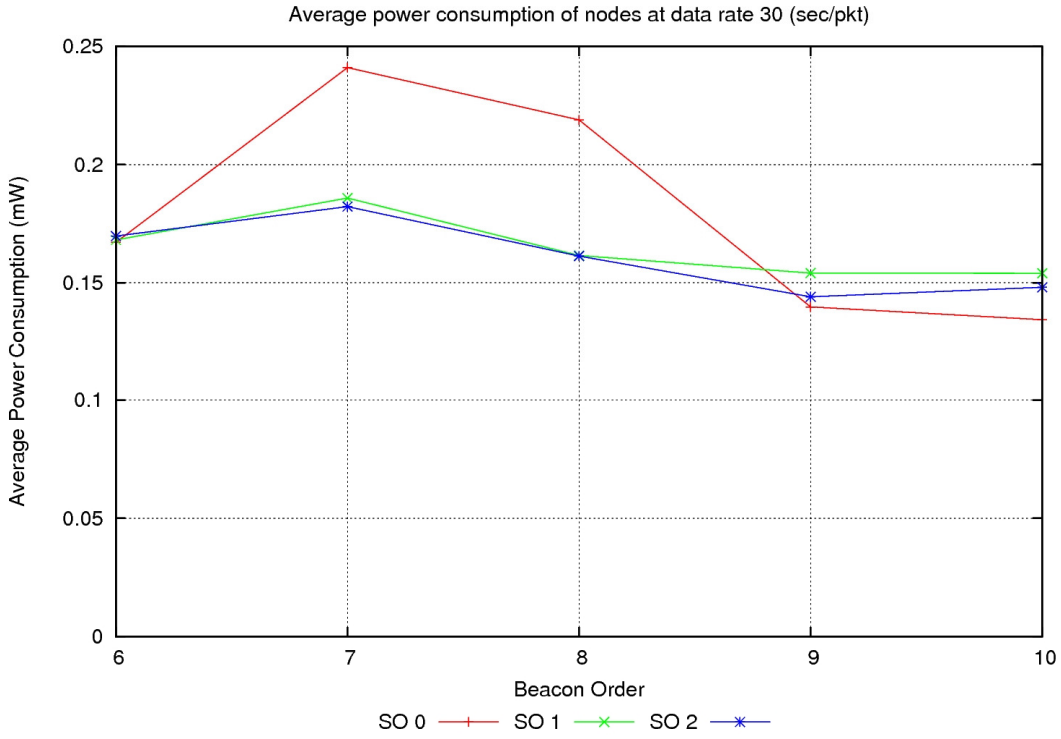


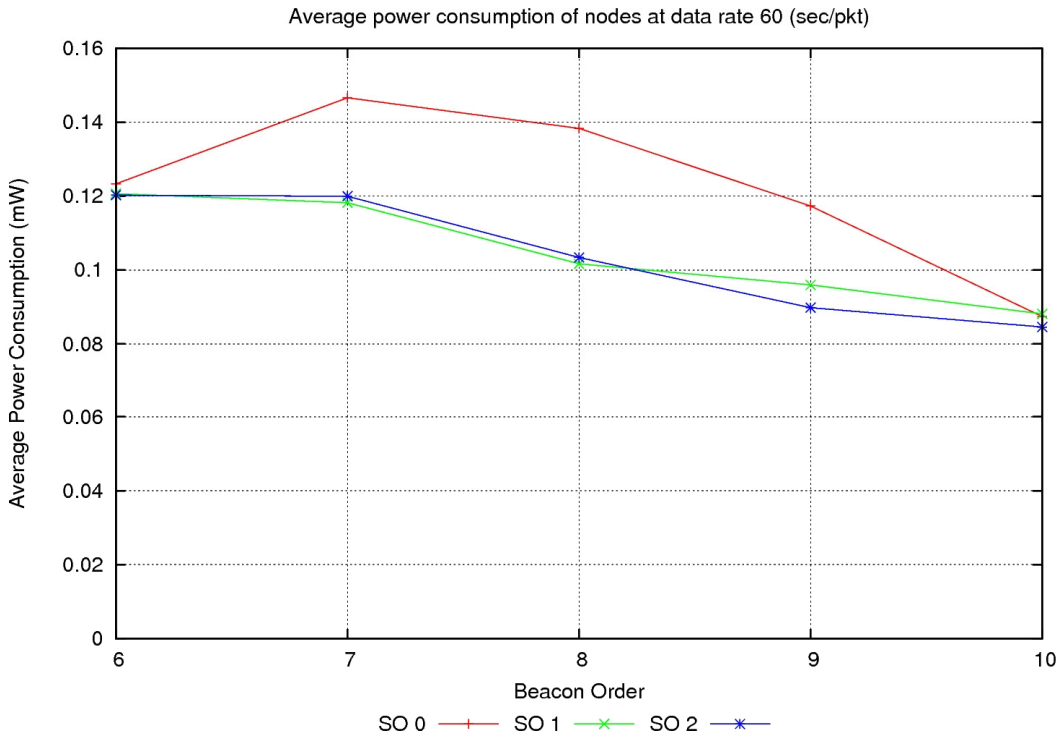
Average power consumption of nodes in non beacon mode

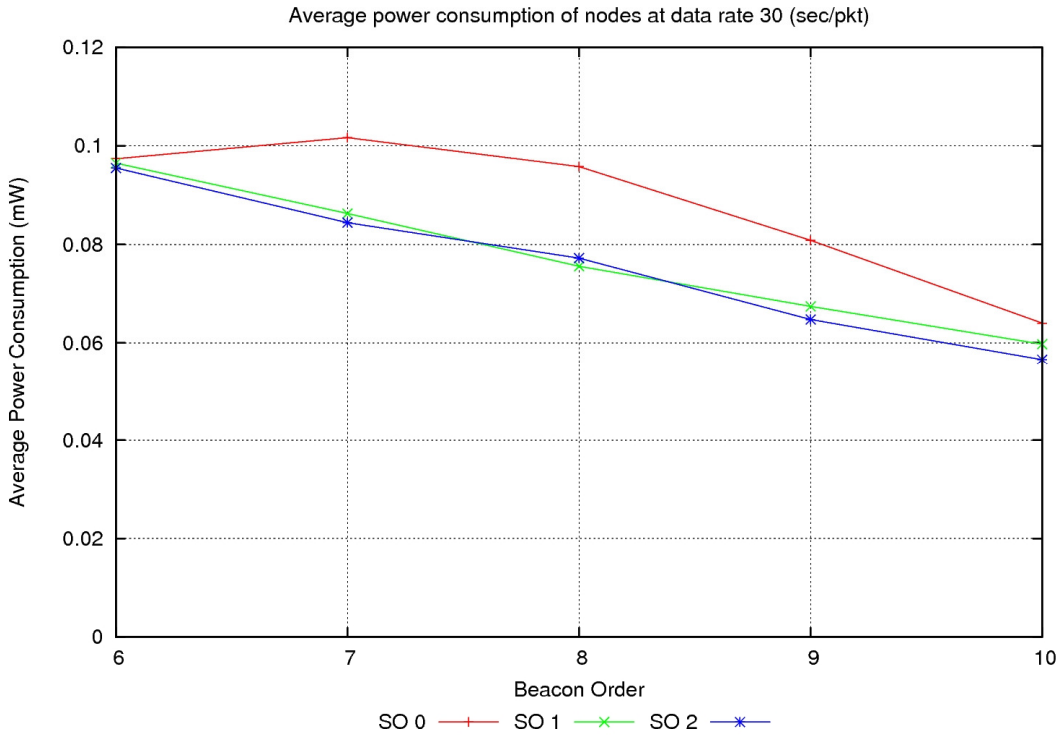


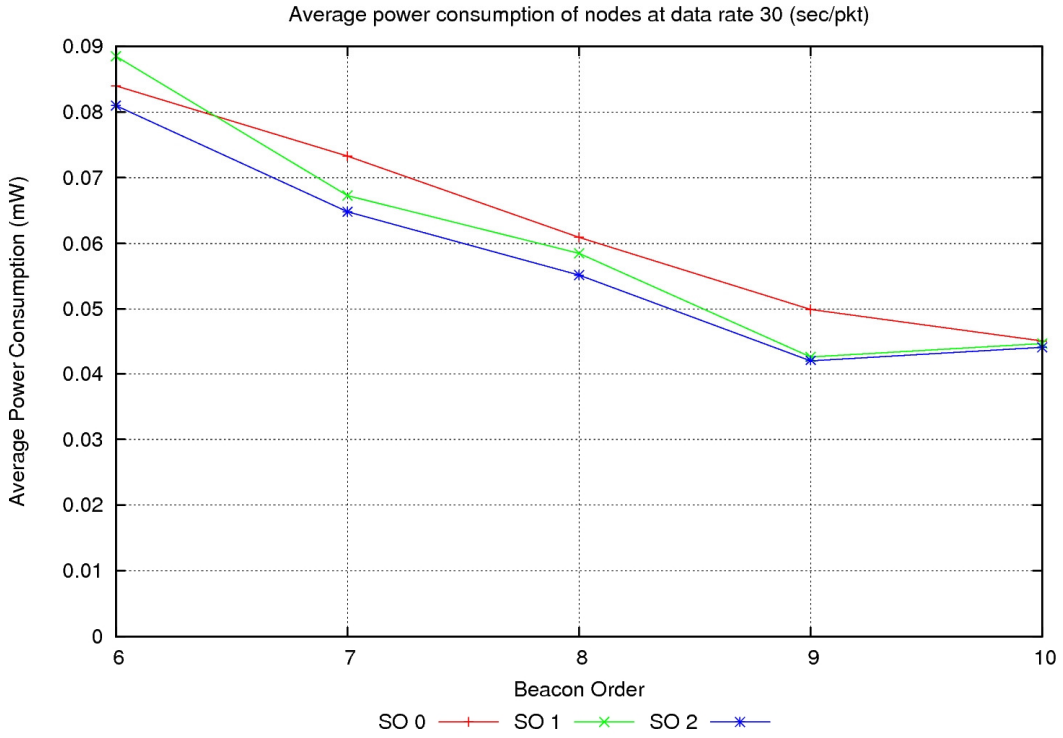
Appendix B - Preliminary Experiments – Without AODV or ARP

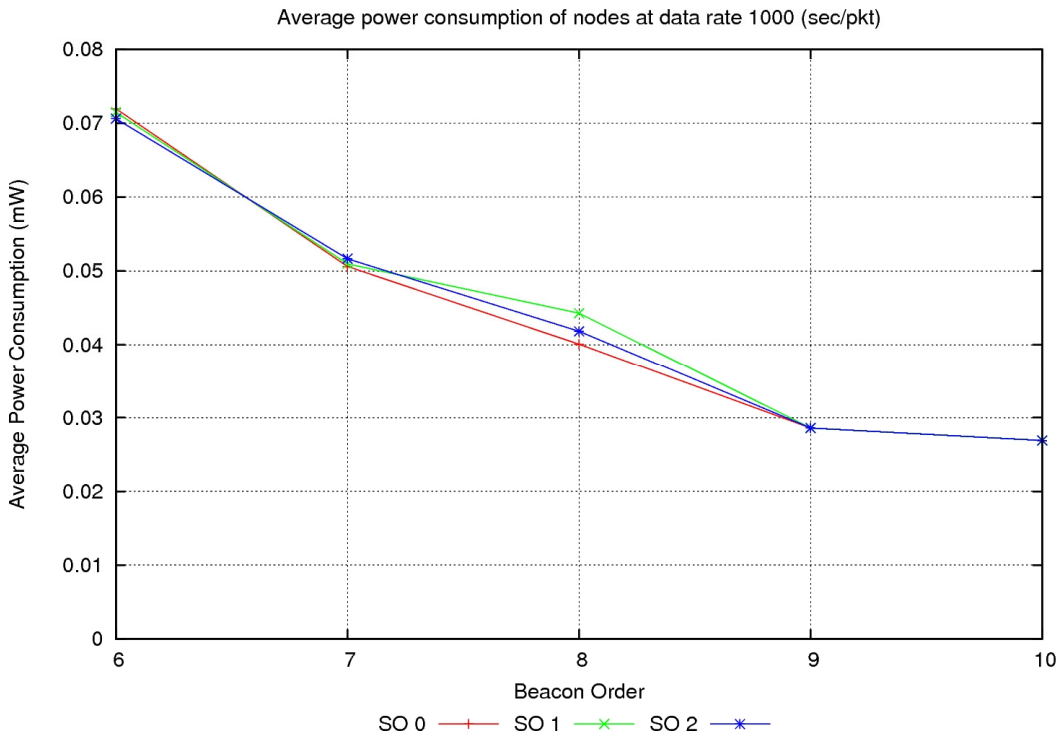
The following graphs are the average power consumption of nodes in beacon mode



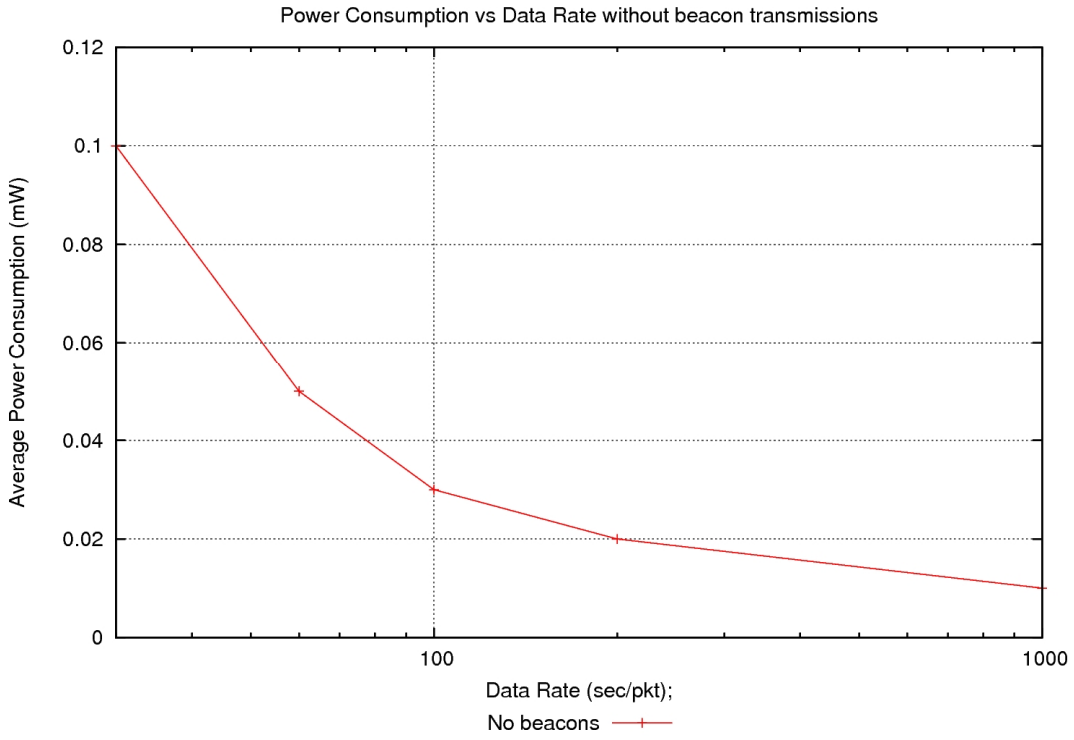



Average power consumption of nodes at data rate 30 (sec/pkt)
Average Power Consumption (mW)
0.12
0.1
0.08
0.06
0.04
0.02
0
6
7
8
9
10
Beacon Order
SO 0
SO 1
SO 2

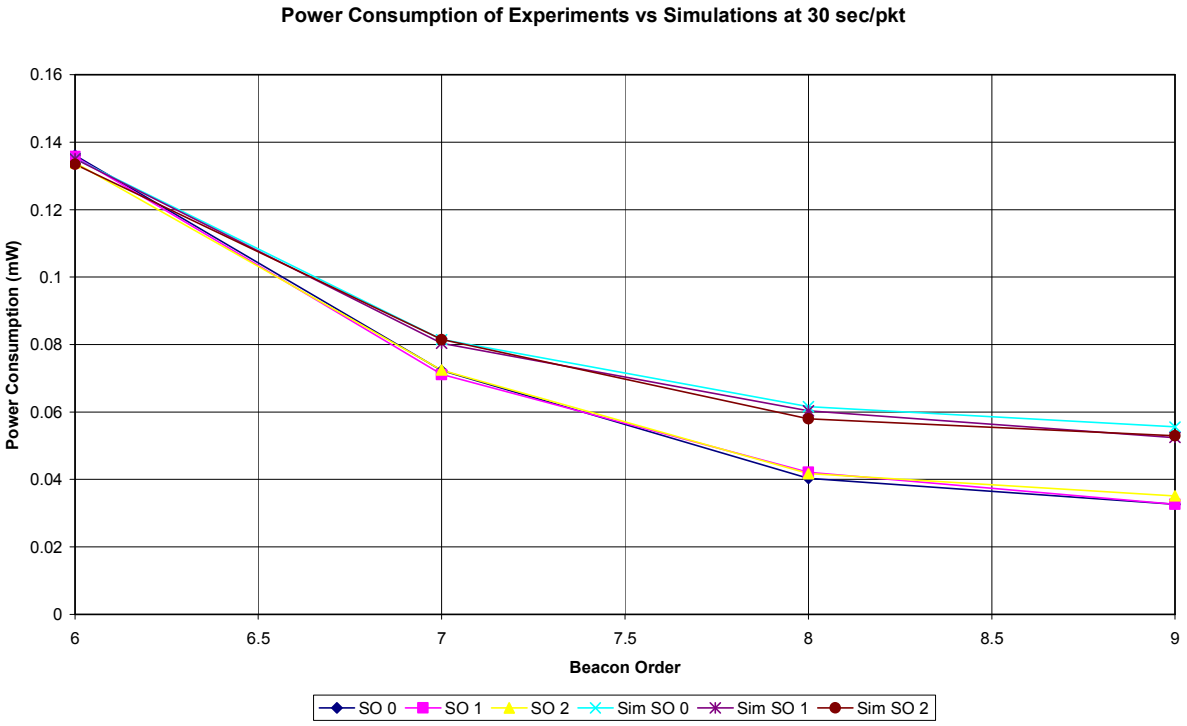

Average power consumption of nodes at data rate 30 (sec/pkt)
Average Power Consumption (mW)
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
6
7
8
9
10
Beacon Order
SO 0
SO 1
SO 2

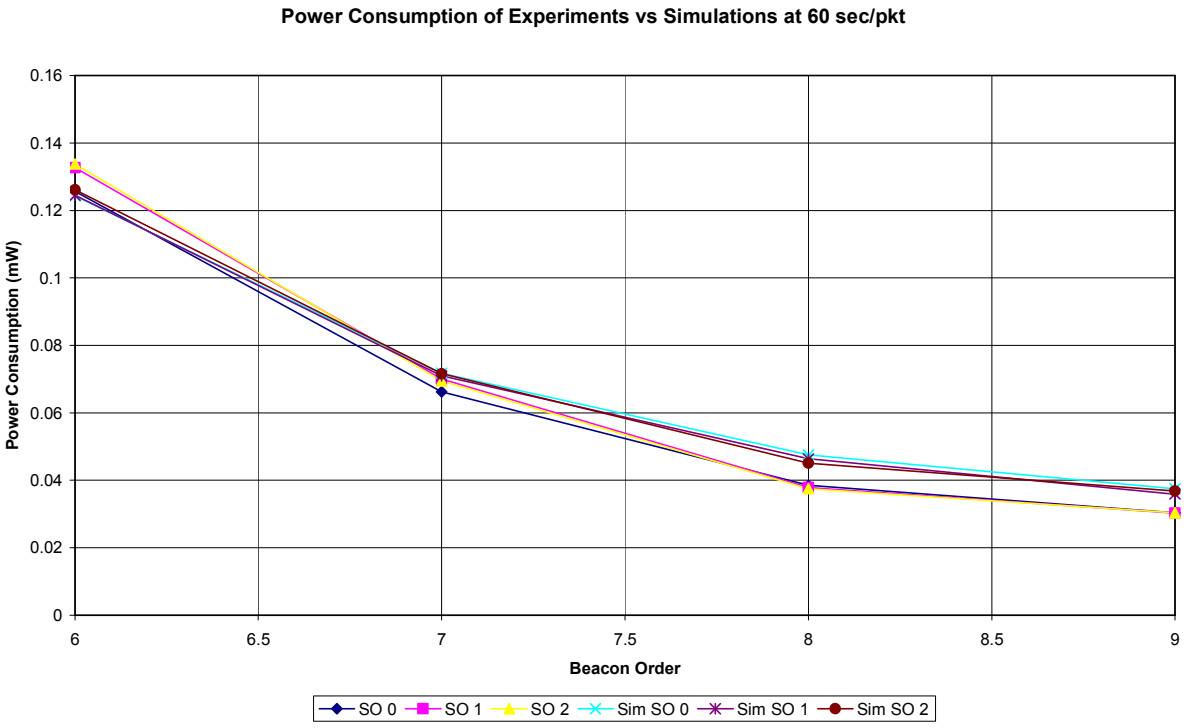


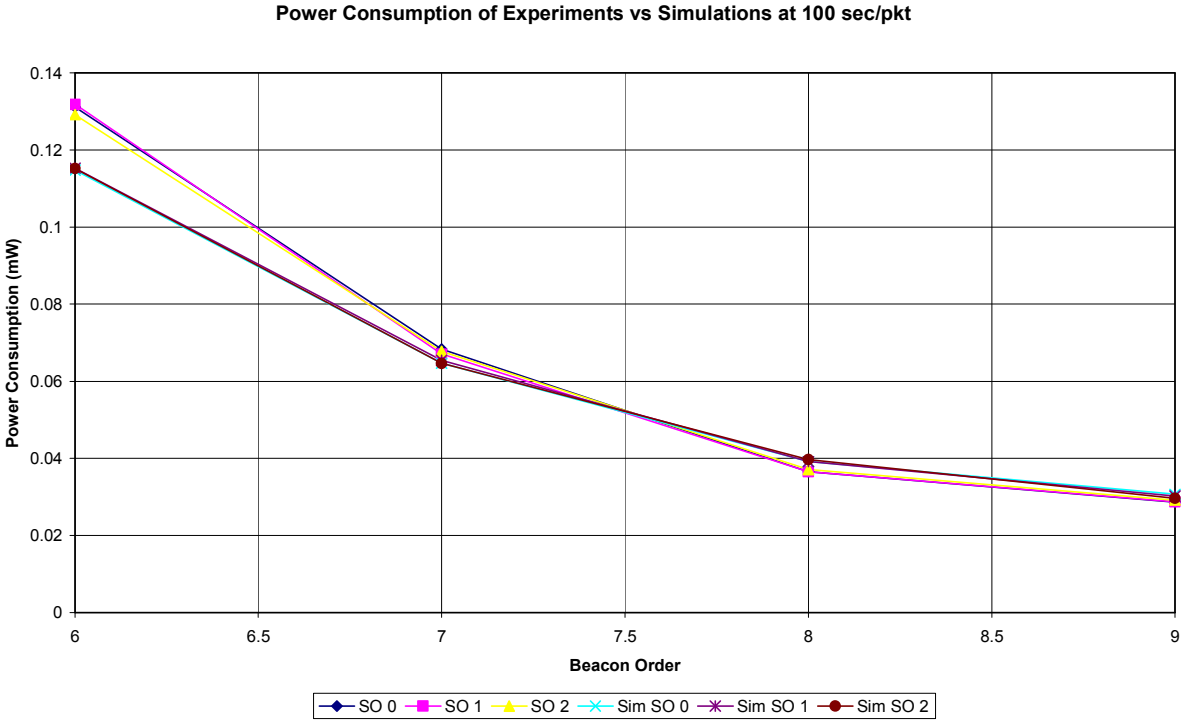
Average power consumption of nodes in non beacon mode

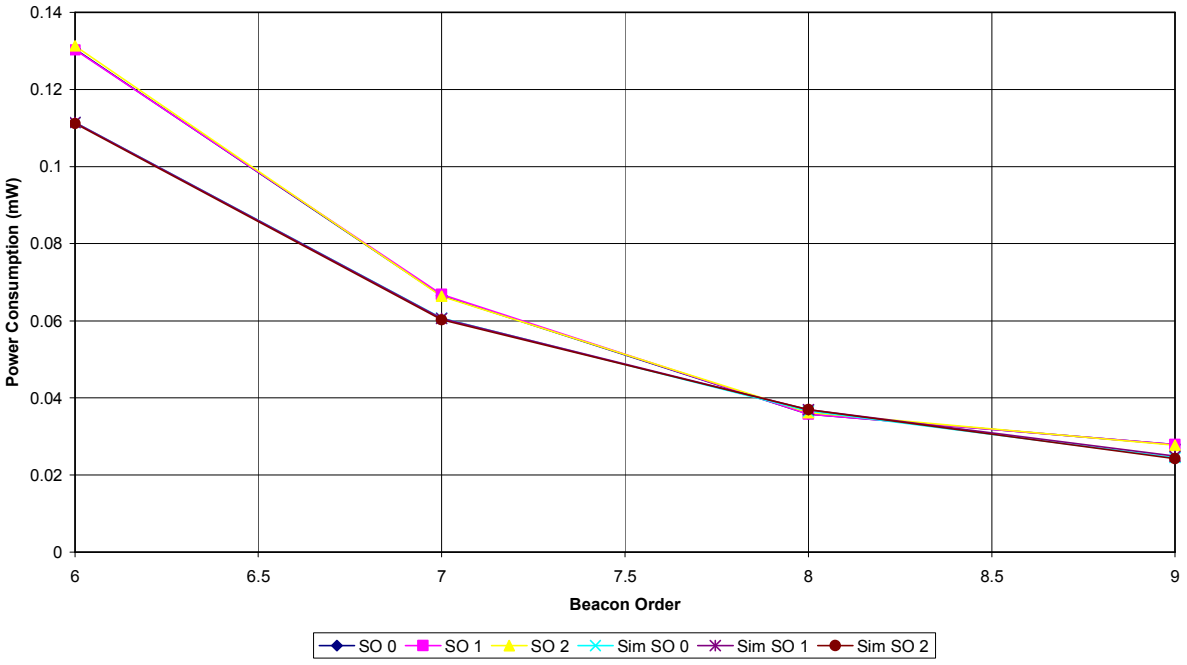


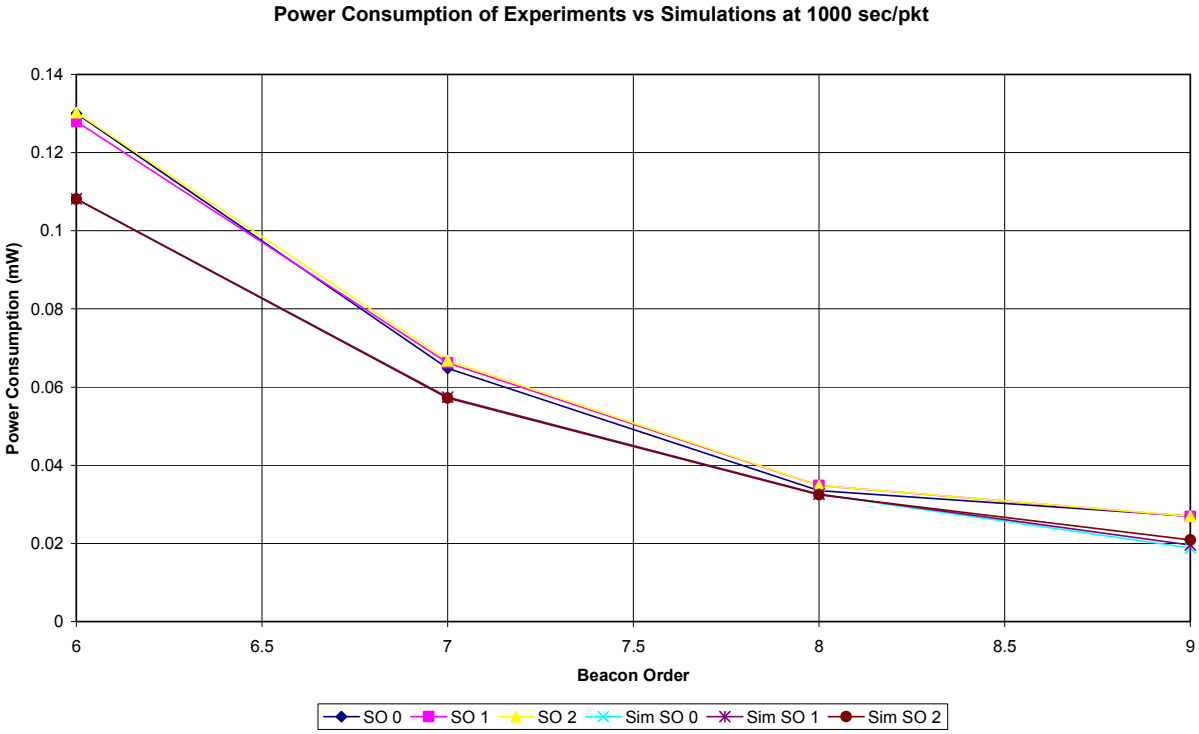
Appendix C – Power Consumption of Final Results






Power Consumption of Experiments vs Simulations at 100 sec/pkt
Power Consumption (mW)
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
6
6.5
7
7.5
8
8.5
9
Beacon Order
SO 0
SO 1
SO 2
Sim SO 0
Sim SO 1
Sim SO 2


Power Consumption of Experiments vs Simulations at 200 sec/pkt
Power Consumption (mW)
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
6
6.5
7
7.5
8
8.5
9
Beacon Order
SO 0
SO 1
SO 2
Sim SO 0
Sim SO 1
Sim SO 2


Power Consumption of Experiments vs Simulations at 1000 sec/pkt
Power Consumption (mW)
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
6
6.5
7
7.5
8
8.5
9
Beacon Order
SO 0
SO 1
SO 2
Sim SO 0
Sim SO 1
Sim SO 2

Appendix D – CD Directory Listing

This CD contains the results from the simulations and experimental results used in this thesis as well as modified simulator files. The file locations are explained below. The software used for the simulations was developed under Linux and thus the files have UNIX line endings and therefore will not display correctly in notepad under Windows. Wordpad however is capable of displaying the files correctly.

The CD is split into two sections, results and source code which are described below.

Results Directory

IEEE 802.15.4 simulation results
Directory: `\Results\Simulation results`

This directory contains three sets of results, the initial simulation results, the results from the modified simulator and the final results for both beacon mode and non beacon mode simulations.

IEEE 802.15.4 experimental results
Directory: `\Results\Experimental results\802.15.4`

This directory contains the experimental results for the IEEE 802.15.4 network experiments.

Zigbee experimental results
Directory `\Results\Experimental results\Zigbee star`
`\Results\Experimental results\Zigbee mesh`

These directories contain the experimental results from the Zigbee network testing in the star and mesh network topology.

Spread sheet and graphs of results
File `\Results\Results.xls`

This spreadsheet contains all the raw data from simulation and experimental results as well as graphs which are presented in this thesis.

Source Code

Modified NS-2 files
Directory `\Source code\NS-2 mod files`

The source code files that have been modified in NS-2 for this research are located in this folder

Simulation scripts
Directory `\Source code\Simulation scripts`

The simulation scripts described in Section 3.8 are located in this folder

Simulation programs
Directory `\Source code\Simulation programs`

The programs used to generate the simulation results as described in Section 3.8 are contained in this directory. These include the `testpower` and `scen_gen` programs.

Packet error rate testing PC application
Directory `\Source code\PER tester app`

This directory contains the source code for the PC based packet error rate testing application

Firmware
Directory `\Source code\Firmware`

This directory contains the four firmware projects for the IEEE 802.15.4 and Zigbee wireless network nodes.

MATLAB data acquisition script
Directory `\Source code\DAQ\`

The MATLAB data acquisition m file script is located in this directory