Hybrid ARQ Based Data Reliability Framework for WSNs using LDPC Codes

By
Umay Kulsoom
2008-NUST-MS PhD-IT-43

Supervisor
Dr. Saad Bin Qaisar
NUST-SEECS

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(March 2012)

Approval

It is certified that the contents and form of the thesis entitled "**Hybrid ARQ Based Data Reliability Framework for WSNs using LDPC Codes**" submitted by **Umay Kulsoom** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Saad Bin Qaisar**

Signature: _____
Date: _____

Committee Member 1: **Dr. Adeel Baig**

Signature: _____
Date: _____

Committee Member 2: **Dr. Anjum Naveed**

Signature: _____
Date: _____

Committee Member 3: **Dr. Zawar Hussain**

Signature: _____
Date: _____

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at National University of Sciences & Technology (NUST) School of Electrical Engineering & Computer Science (SEECS) or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Umay Kulsoom**

Signature: _____

To my parents

Acknowledgments

First of all I would like to express my deepest gratitude to ALLAH The Almighty. Its only the belief in Him that makes it possible for me to complete my thesis. I can never enough acknowledge Him as being an ordinary human being, but all what I have achieved in my life till today is just His blessing.

I would like to express my gratitude to my supervisor, Dr. Saad Qaisar, whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate his vast knowledge and skill in many areas and his assistance in writing reports, which have on occasion made me "GREEN" with envy.

I would like to thank the my committee members for the assistance they provided at all levels of this thesis.

Special thanks to my friends and lab mates. You people have always been there for me and and the support that I get from you people is just marvellous.

Last but not the least this would not have been possible without my parents. My parents have provided me immense support, love and care throughout my life and in particular during MS thesis. Thanks Allah for blessing me with great parents.

Umay Kulsoom

Contents

List of Figures

List of Tables

Abstract

Reliability is the most important feature of event-driven wireless sensor networks. Due to the resource constrained nature of sensor networks, achieving reliability is challenging with efficient utilization of communication resources. Designing energy-efficient data communication mechanisms plays a vital role in maximizing the life-time of wireless sensor networks. Since communication is accomplished through lossy error prone wireless link where errors are introduced in a packet while transmission from the source to destination. Motivated by above mentioned challenges, we propose a distributed HARQ based data reliability framework for reliable data delivery adaptable to varying network conditions. Proposed error-correction framework is capable of transmitting data reliably with least amount of retransmissions. Our work is based on the fact that rate of energy depletion is higher in transmission and reception as compared to processing at sensor CPU. We integrated Low Density Parity Check (LDPC) codes in order to achieve high level of reliability in distributed fashion for efficient data communications. Intermediate relay nodes perform certain degree of processing in order to partially recover corrupted data. Such approach has significant impact on the lifetime of network and helps in reducing the bit error rate at destination. Framework is implemented using Crossbow Imote2. Actual on-sensor implementation of the proposed and traditional techniques enabled us to show the efficacy of our framework with fair comparison.

Chapter 1

Introduction and Motivation

1.1 Introduction

Wireless sensors are battery powered self configuring devices that collaborate with each other and form an adhoc multi-hop wireless network. Technological advancements and miniaturization of electronic devices have made Wireless Sensor Networks indispensable in our practical lives with wide range of applications from security to health care. Indeed development and high integration of WSNs in our daily lives enhances human capabilities regarding data collection and monitoring of inaccessible areas even. The observations could be light intensity, temperature, pressure, sound intensity, images, and so on. WSNs are generally placed in human inaccessible environments without any fixed power sources. They are therefore required to last on the order of years on power sources such as batteries, solar cells, vibrations, and thermo-electric effects [22]. These factors raise issues not only for hardware design but also in the approach taken when designing the protocols and algorithms for these devices. Most of the design strategies for networks such as wireless LANs do not work effectively for WSNs due to the inherently different design constraints. In spite of application targeted by WSN, ultimate goal is 'reliable data delivery' accomplished through unreliable and dynamic error prone wireless links. Typically, WSNs are deployed in two types of scenarios: Single-hop and Multi-hop. In Single-hop communication, the information sink or the base station is within one hop transmission distance of source node. In Multi-hop topology, there are multiple communication hops between the sensor and the base station. Intermediate sensors at each hop act as routers which forward packets from a sensor towards the base station. However traversal of data through different nodes increases data corruption. Reliability has remained one of the challenging issue and an open research

area.

'Energy' is one of limited resource and requires to be utilized intelligently for prolonged network lifetime. The two main sources of energy depletion on a wireless sensor are local computations and bit transmissions/receptions. A generally accepted fact is that energy required for one bit transmission/reception by a sensor's radio significantly exceeds the energy required for local computation on a sensor's CPU [21].

Design of sensor network is mostly application specific and reliability requirements vary with the scope of application. For instance applications that collect and send data periodically like temperature, habitat monitoring can tolerate loss and corruption in data. However, event-driven networks got critical information pertaining to the event and requires reliable transmission to the central node or base stations. Such applications where data of each node is important and critical information has to be transferred reliably and maximum error free are the main motivation of this thesis to come up with techniques of reliable communication.

1.2 Key Contributions

Present work proposes communication protocol to deliver data adaptable to network conditions and aims to provide maximum reliability with minimum energy consumption. Main contributions of thesis work are as follows:

- We developed insight into reliability requirements of sensor network applications where individual node data is important and maximum data reliability needs to be achieved. Since maximum reliability is possible at the cost of high energy consumption so we worked out to find a trade-off between providing maximum data protection at reasonable energy usage.

- We have proposed Hybrid ARQ based framework for reliable data transmission in the network adaptable to dynamic changes taking place. Our basic objective energy efficient communication; vital for the lifetime maximization of wireless sensor networks is achieved through transmission of data with minimum retransmission and hop by hop reliability is provided. We have used Low Density Parity Check (LDPC) codes to protect data from corruption and loss is informed to source node by using negative acknowledgements.

- Error detection mechanism has been proposed at the intermediate nodes making them intelligent in decision making for forwarding and/or pro-

cessing of data. Processing at relay nodes has been proposed to minimize the number of errors induced in the packet and reducing retransmissions. We exploited the iterative nature of LDPC codes to detect error and to correct them.

- Proposed protocol has been evaluated using Crossbow Imote2 designed around the low-power PXA271 XScale micro- controller with 802.15.4 radio and surface mount 2.4 GHz antenna. Experimental evaluation of proposed protocol has been carried out indoor environments.

1.3 Thesis Organization

Rest of the thesis is organized as follows.Chapter 2 provides a brief background and reviews the related work. Data reliability framework has been explained and discussed in chapter 3. In chapter 4 we summarize the performance evaluation of our proposed framework. Finally, we conclude in Chapter 5 with the list of possible future extension.

Chapter 2

Literature Review

Background

2.1 Error Control

Unreliable and error prone lossy wireless links induce errors in the packet corrupting data and making it useless for receiver. To achieve reliability we need some error control technique to protect our data.

2.1.1 Backward Error Control: Automatic Repeat Request

Automatic repeat request (ARQ) ensures that data stream is delivered accurately to the user despite of transmission errors [13]. Stop and Wait (SW), Go-back-N (GbN) and Selective Repeat (SR) are three types of ARQ. SW is most simplest of all and data delivery progresses packet by packet with acknowledgement of each. If packet is not acknowledged in a particular time slot, packet is retransmitted however if packet contains erroneous data then there is no mechanism for error correction at receiver end. GbN in contrast to ARQ,transmits a stream of packets number specified by window size. When maximum window size is approached and there are outstanding packets that have not been acknowledged then GbN would start retransmission from the packet that has not been acknowledged despite the fact the preceding packets are acknowledged. This feature made GbN infeasible for highly error prone links. To counter unlimited retransmission, SR selectively retransmits the missing packets. However it requires large size of buffers available at the source to hold the transmitted packets making it inappropriate for sensors. Therefore SW-ARQ; inherently simple is ideal for wireless sensor networks

when channel error rate is not high.

2.1.2 Forward Error Correction(FEC)

Forward Error Correction adds up redundancy in the packet to protect data from corruption. Careful selection of redundancy allows receiver to detect and correct limited number of message bits without requiring any retransmission from the source. FEC allow us to protect data either by compressing data at source node known as source coding or by adding redundancy to data to make noisy channel appear as noiseless. Channel Coding helps in catering the noise inherent in wireless links. Simple channel coding schemes allow the received of the transmitted data signal to detect errors, while more advanced channel coding schemes provide the ability to recover a finite about of corrupted data.

Although channel coding provides reliability and cutting down the retransmission requirement however there is an overhead with transmission of extra bit.

2.1.3 Hybrid ARQ

Both ARQ and FEC have their own advantages and disadvantages in different network conditions. Wireless sensor networks operate in varying network conditions therefore we need to have such solution that take advantage of both schemes while intelligently conserving energy and providing reliability. *HybridARQ* is such a tradeoff between ARQ and FEC that functions on top of benefits of both schemes. There are two types of HARQ schemes; Type I and Type II. HARQ-I allow us to first transmit an uncoded packet or a packet coded with a lower error correction capability. If this packet get corrupted and negative acknowledgement(NACK) is received then packet is re-sent with a powerful FEC code. In Type-II only redundant bits are retransmitted for successful recovery of packet at receiver. Type I eliminate the buffer / packet storage requirement at sender however Type II reduces bandwidth usage. We intend to use Type I Hybrid ARQ in our thesis

As mentioned above sensor networks may face time varying links therefore if we make Hybrid ARQ channel aware then it not only helps in decision making regarding coding rate (amount of redundancy added to the packet) but also helps in reducing retransmission(s).

2.1.4 Link Quality Estimation

Wireless communication is hampered by numerous factors for instance harsh environments where devices are operating in, interference, shadowing, multi-path effects etc. Simultaneous transmission by the devices coexisting in the channel cause interference while attenuation, scattering, reflection and obstacles are the reasons for multi-path fading and shadowing. Strong signal requires higher transmission power!. Higher transmission power means strong transmitting signal and fewer errors yet sensors are intended to be deployed for a longer period of time and frequent battery replenishment is not possible therefore they need to conserve energy while transmission. Quality of radio links available in Wireless Sensor Networks experiences dynamic variation over time and space thus making communication over them extremely unreliable. Therefore, carrying out reliable communication in such networks require transmissions to be adaptable to network conditions. Link quality estimators are link quality measurement metrics which help us in determination how good communication links are. Several Link Quality Estimators (LQEs) [5] have been reported in the literature; broadly categorized as receiver side estimators, sender side estimators and hybrid side estimators. Prominent receiver based estimators are:

1. Received Signal Strength Indicator (RSSI): is the strength of a received RF signal.

2. Signal to Noise Ratio (SNR): expressed in decibel (dB) is a hardware metric that quantifies how much a signal has been corrupted by noise. It is the ratio of signal power to the noise power corrupting the signal.

3. Link Quality Indicator (LQI): is a measure of chip error rate.

4. Packet Reception Ratio (PRR): is the ratio of packets received to the number of packets sent. Lost and received gives the total number of sent packets while packet losses at receiver end are determined with the gap in sequence numbers and counting them.

In such situation we need such packet delivery mechanism that not only ensure successful packet delivery but also protect and correct errors induced in packet

2.1.5 Low Density Parity Check Codes

Low Density Parity Check codes (LDPC) [8] are class of linear block codes introduced by Gallagar in 1963 and are proven to be most efficient codes for

Wireless Sensor Networks. These codes are asymptotically good and perform close to Shannon capacity as block length increases.

2.1.5.1 Encoding

Generator Matrix Representations

2.2 Related Work

Reliability is important for every kind of WSNs applications but critical for health and military applications. Some applications are loss- tolerant and require a particular ratio/level of reliability. DTC [30], TSS [9] , PSFQ [1], RMST [2], RCRT [27] provide 100% reliability for most of the wireless sensor network applications. However, these protocols do not provide same mechanism for loss-detection and recovery. ESRT [29], DTSN [31] and SCTP[28] provide classes of probability for the application.

SCTP [28] and RCRT [27] utilize sequence numbers at the receiver end for loss-detection and retransmissions are carried out from source node. These protocols provide end-to-end recovery, however this approach is not energy-efficient and new protocols enable intermediate nodes to cache segments. Caching segments at intermediate node reduces the cost and number of total exchanged messages between source and receiver. Two kinds of intermediate nodes are used by most of the protocols. First does not detect losses but reacts when it receives a NACK message by retransmitting missing segment. The second kind detects losses and requests a retransmission from its neighbours.

TSS [9] and ERTP [32] use implicit acknowledgement (IACK). This mechanism requires that each node i after sending a packet the next node to the sink overheads the next forwarding. The forwarding of packet by node i+1 is considered as an acknowledgement to the node i.

Flush [21] is a reliable single-flow bulk transport protocol for large diameter WSNs. However, Flush only supports one data flow using an end-to-end approach robust to node failures. Flush requires that the sink node sends the sequence numbers of packets it did not receive back to the data source. When a source node receives a NACK packet, it retransmits the missing data. Flush proposes also a rate allocation scheme for adapting dynamically the sending rate of the sensor nodes while considering the broadcast nature of medium and the interference between nodes. The rate allocation algorithm follows two basic rules: 1) Rule 1: A node should only transmit when its successor is free from interference. 2) Rule 2: A node's sending rate cannot exceed the sending rate of its successor. These two rules reduce contention

and thus collision in the wireless network and minimize losses due to the queue overflows for all nodes.Flush is compared to fixed rate algorithms and it was seen that Flush provides more reliability and a better average throughput. However, Reliability is not given in numbers and therefore its not very much that how much reliability is provided. Energy Efficiency of proposed algorithm is also missing

[12] Considers rayeligh fading channel and proposes channel aware ARQ protocol. They have used probing protocol with time interval for the transmission of probing packet dependent on length of fading. The protocol show improvement with channel awareness and use of probing protocol however pilot and data packets are of same size resulting in increased overhead plus comparative faster depletion of energy

[25] has proved that BCH codes are 15% more energy efficient than best performing convolutional codes. Moreover thy have shown that LDPC codes provide reliable communication and are 42% more energy efficient than BCH codes

Joint source and channel modelling for WSN has been frequently investigated by researchers. Hasan et. al [13] has exploited the advantages of joint source and channel coding particularly related with aggregation and security of data in WSNs, with primary focus on source coding. However end-to-end channel coding is used, which leads to relative high draining of sensor nodes power.

A work related to ours is found in [26] where Radha et. al have proposed a framework using partial processing in the network. They propose to use LDPC codes to perform certain processing i.e. partially decode the received packet and then forward the processed packet. Such processing results in significantly reducing the error rate in packets however, this processing has not been made compatible with network conditions which may cause overhead and high use of scarce resource when channel quality is comparatively better!

Chapter 3

Problem Formulation

Energy efficiency plays a vital role in maximizing the lifetime of wireless sensor network. Limited battery constraint dominates lifetime and efficiency of almost all kinds of WSN application. Wireless channels are highly prone to errors and the error rate on these links accumulates exponentially as network size become large[11]. Therefore, maximum error free data transmission to the destination node is of high importance to meet the target of WSN application. However maximum reliability is achieved at the cost of high energy consumption. So, in order to transmit energy efficiently we need to protect our data from errors introduced in a packet while transmission through wireless link. Also, large number of retransmissions result in faster depletion of energy. So we need to design a protocol for wireless sensor that can deliver data 'reliably' and 'energy efficiently'.

In this thesis we evaluated

> Can we design a HARQ Protocol for wireless sensor networks that performs better than ARQ and LDPC in delivering data reliably and energy efficiently?

Protocol designing for wireless sensor network require careful consideration of various factors from data generation to transmission. Starting our discussion from source node, ideally we want that source node transmits maximum data for which we need large packet size. Higher number of errors would be introduced by the erroneous wireless link(s) traversed from source to destination resulting in retransmission(s) costs high in terms of energy consumption. If small packet size is used then packet error rate would be less however there would be high packetization overhead. Similar to packet size, another problem is the selection of optimal transmission power. In case of high transmission power, lower error rates are achieved but depletion rate

of battery of a node is high. Lower transmission power results in saving of battery but higher error rate is the resultant. So we need a trade-off between large and small packet size as well as high and low transmission power. Once packet is ready for transmission next thing that comes is channel. We are well-aware that wireless links are highly unreliable and dynamic so next consideration is that do we need an estimation of channel quality before transmission or not? Talking from data reliability perspective, maximum possible error free data transmission is required. In order to protect data from the corruption introduced by wireless links; either we may like to have a backward error control (ARQ) where each or a bunch of packets is acknowledged by the receiver or forward error control (FEC) where redundancy is added to data to protect it. For backward error control we need feedback mechanism; playing a major role in providing reliability but at the cost of some extra communication. Another issue in feedback based schemes is in the selection of decision-making nodes thus providing feed back to the sender. Most of the time, providing feedback is the responsibility of destination node i.e. end-to-end approach however its energy consuming and packet loss is high when wireless links are highly error prone. Some times, intermediate nodes provide acknowledgement of each packet or cumulative ACK/ NACK for the bunch of packets. Forward error control does not need any feedback mechanism but redundant data transmission costs high when network is operating under favourable condition. Therefore, code rate selection is an important issue while dealing with FEC. Its a well known fact that communication cost is higher in sensor networks as compared to computations and processing performed at node local CPU. So we need to figure out that whether it is beneficial for us to do some processing on the packet as compared to as-is forwarding of packet!

Based on above discussion following questions need to be answered carefully while designing data reliability protocol for wireless sensor networks:

1. What should be data transmission mechanism under different network conditions?

2. Do 'channel awareness' helps in achieving reliability?

3. What should be the feedback mechanism? Are Implicit Acknowledgements good enough to ensure reliability or Explicit acknowledgements are required?

4. What kind of error recovery mechanism between end-to-end and hop-by-hop provides reliability at reasonable energy consumption?

5. How the code rate (amount of redundancy added to packet to protect data) should be defined?

6. Is it efficient to make code rate adaptive to channel quality?

7. Is it beneficial to make the intermediate node(s) intelligent regarding processing and/or forwarding of packet?

Chapter 4

Proposed Methodology

For reliable data communication at optimal energy consumption we followed Hybrid ARQ based approach to deliver data from source to destination. For doing so, we opted methodology stated and explained in this chapter.

4.1 Hybrid ARQ based Data Reliability Framework

We have categorized nodes in our network as source nodes, rely nodes and destination node. Assuming that network is formed and each node is equipped with its routing table having knowledge of number of nodes required to be traversed to reach the destination / base station. Nodes are considered to be placed in line topology with variable distance between them

4.2 Error Control Mechanism

Errors inherent in wireless communication corrupt data in a packet and hence make it useless for receiver. Since energy consumed by radio circuit in transmission or reception of 1 bit is much higher than the processing of bit at the node. Considering this fact we have proposed a data reliability framework whose key features are:

- Hop by Hop Reliability
- Link Quality Estimation
- Buffers
- Partial Processing

Two main types of error control codes used in an FEC system are block codes and convolutional codes [4,9,11]. [25] proves that BCH code are 15% more energy-efficient than the best performing convolution codes proposed to date for WSNs. Block codes are also easier to implement on resource-constrained devices and the encoding energy required for them can generally be considered negligible [17]. FEC helps in more reliable communication since they increase the effective transmission range of a node as compared to ARQ using same transmission power and eliminates retransmission.

4.2.1 Hop by Hop Decision Making

Wireless sensor networks are mostly deployed in multihop fashion to transmit data from source node to destination. Data packet traverses several nodes in the network hence errors introduced in packets influence the selection of error control mechanism. To date several mechanisms have been proposed to overcome packet losses over the harsh channel broadly categorized as end-to-end (E2E) and hop-by-hop (HBH) schemes [1, 2]. HBH considers 100% reliability and overall energy consumption is less as compared to E2E where retransmission takes place all the way from the source node to the destination node.

Frequency of retransmission(s) is reduced by ensuring successful delivery of data hop by hop. Present work relies on the fact that if packet get corrupted at one of the initial hops then energy expenditure would be in transmission of packet from source to a specific node which has received corrupted packet as compared to complete retransmission all the way from the source to destination [15]. So, energy could be conserved providing an opportunity to use adaptive scheme. Strong error control technique is used for packets travelling more hops and weak error control for packets with fewer hops primarily depending on channel quality.

Retransmission limit has been set to 3 in proposed HARQ. If packet is unable to be recovered in three retransmissions it would be discarded instead of further retransmissions resulting in higher depletion of battery.

4.2.2 Link Quality Estimation

Protocol efficiency is highly improved with link quality estimation. However, improvement in efficiency is closely tied up with the accuracy of link quality estimates.

Imote2 uses CC2420 radio which provides RSSI and LQI. RSSI which is the estimate of received signal power is calculated over 8 symbol periods and

stored in the$RSSI_VAL$ register. Chipcon specifies the following formula to compute the received signal power:

$$P = RSSI\_VAL + RSSI\_OFFSET$$

where: $RSSI\_OFFSET$ is about -45 dBm.

LQI; chip error rate is calculated over 8 bits following the start frame delimiter (SFD) and its values are usually between 110 and 50 corresponding to maximum and minimum quality frames respectively.Computation of LQI is vendor specific. Therefore, we are using RSSI as link quality indicator metric in our framework.

Code rate; amount of redundancy added to the packet depends on RSSI value. Source node picks up the coding rate in accordance to channel quality explained in detail later.

4.2.3 Buffers

Buffers are used to achieve the desired reliability holding failed packets for a small time interval. Due to limitation of sensor hardware, it cannot be assumed that size of available buffers is infinite [12]. Thus, we need to quantify the effect of this limited buffer on the performance of proposed framework. If link state appears to be 'good' frequently, we may not need a large buffer. Contrary to this if 'bad' state continues for a long time, we have to have a big buffer.

4.2.4 Partial Processing

Energy consumed by a sensor node in transmission and/or reception is much higher as compared to local processing and computation. Taking advantage of this fact, we propose to perform some computation for accurate decision making about retransmission of the packet. Packet forwarding and/or requesting for retransmission is dependent on the processing performed by As we know LDPC codes are iterative in nature so relay node are required to perform 1 iteration; subject to energy availability and channel quality.

4.2.5 Error Detection Mechanism at Relay Nodes

Following figure gives an overview of the proposed protocol:

Assuming that network has been formed and all links are established therefore each node is equipped with its routing table (of the form shown

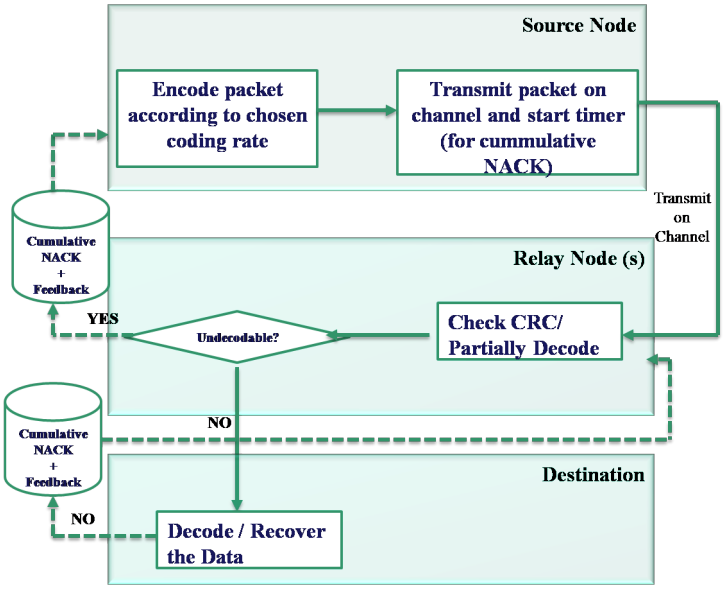

Figure 4.1: Overview of Proposed Solution

in following table) and RSSI value through the exchange of beacon packets during routing phase.

Table 4.1: Routing Table

Address	Hops to Base	Next Hope
1001	2	1002
1002	1	1003

An event is sensed by an source node who is aware of the channel quality. Based on channel quality source encodes the data and sends it over to the wireless link. Data is received by the relay node. In case the channel quality is sufficiently good, relay nodes checks CRC and forwards the data to the next node.

If the channel quality is worst which is mostly the case then relay node partially decodes the data i.e. it checks its energy found sufficient enough then it performs 1 iteration on the packet. After one iteration, percentage bit change is detected by the decoder. Percentage bit change is the number of bits required to change to retrieve original code word. Based on the routing table, we know the number of hops to the destination and next hop; so decision regarding the forwarding or retransmission is made as follows.

If *bit_change* is the number of bit change detected by the decoder and

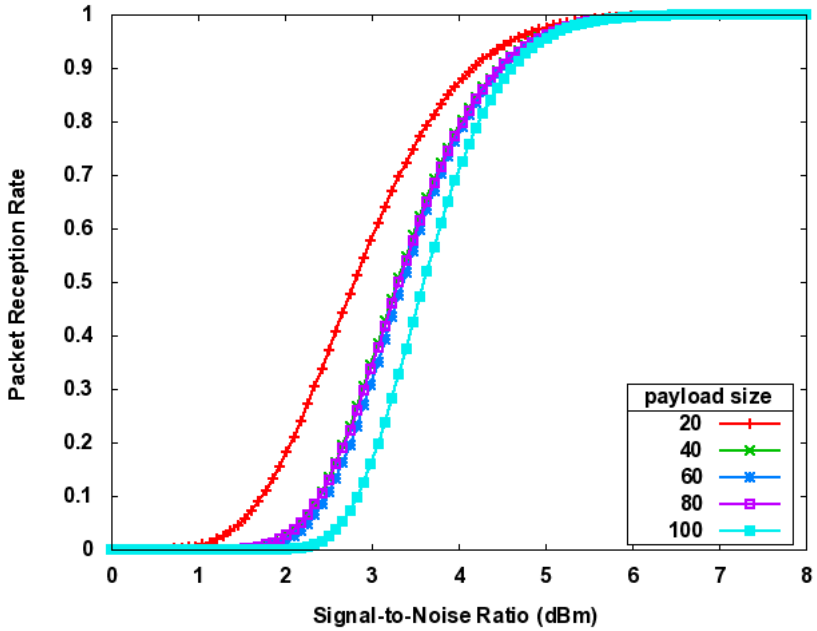

Figure 4.2: SNR

no_hops_destination are the number of hops that have to be traversed yet by a packet, then we adapt the following criterion to decide either to make request for retransmission or forward the packet:

$$retr\_var \leq \frac{bit\_change * no\_hops}{N}$$

where N is the total number of nodes in network.

When *retr_var* satisfies above criterion, the relay node does not request for retransmission of received packet and transmits it to its next node. In contrast, the relay node request retransmission when X does not satisfy this criterion. If channel quality is good the load of relay nodes decreases because the relay nodes will not use decoding process and decision is made on the basis of CRC. It is pertinent to mention here that if any of the relay node does not have sufficient energy to carry out decoding process then partial decode and detection would be carried out at next node.

Protocol for Rate Adaptation Based on Link Quality

Source node start of data transmission start with a higher data rate depending on channel quality. If channel quality degrades with time detected by

the sequence of failed packets, increase number of redundant bits and reduce number of data bits in the packet. However increase data bits when channel quality with higher RSSI is reported

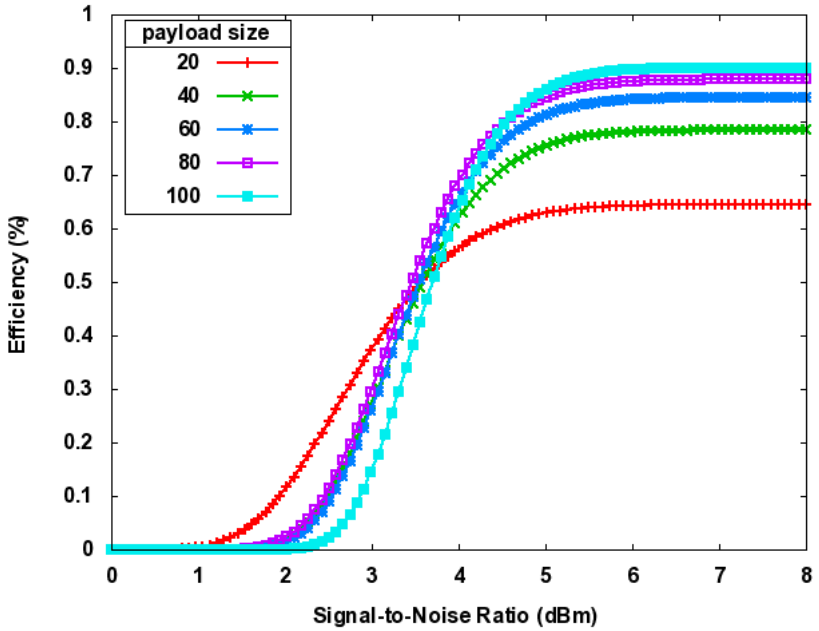

Figure 4.3: Efficiency

Receiver appends the RSSI value when sending NACK back to sender.

From Figure 4.2, following observations are made on the relationship between data bits in a packets and SNR versus PRR. Packet of size 100 bytes is assumed with varying number data bits i.e. for instance 20 bytes actual data and rest is considered redundancy:

- The difference in PRR occurs in the SNR range of 1 dB to 6 dB. For SNR $\leq$ 1 dB, PRR is 0 for all packet size; for SNR $\geq$ 6 dB, PRR is almost 1 for all packet size. While restricting ourselves in the same SNR range; we can say
- For same SNR, lower data size has higher PRR.
- The difference in PRR increases as the SNR decreases. For example, PRR at SNR 5 dB is almost the same (about 0.98) for both 20 and 100 bytes data size. On the other hand, at SNR 2.5 dB, PRR for 100 bytes payload size is almost 0, while the PRR for 20 bytes payload size at the same SNR is about 0.4.

From Figure 4.3,we observe that payload size 100 bytes has the best efficiency atSNR $\geq$ 4.5 dB; 80 bytes payload size has the best efficiency in SNR range 3.5 db $\sim$ 4.5 dB; and 20 payload size has best efficiency in lower SNR range.

4.3 Energy Consumption

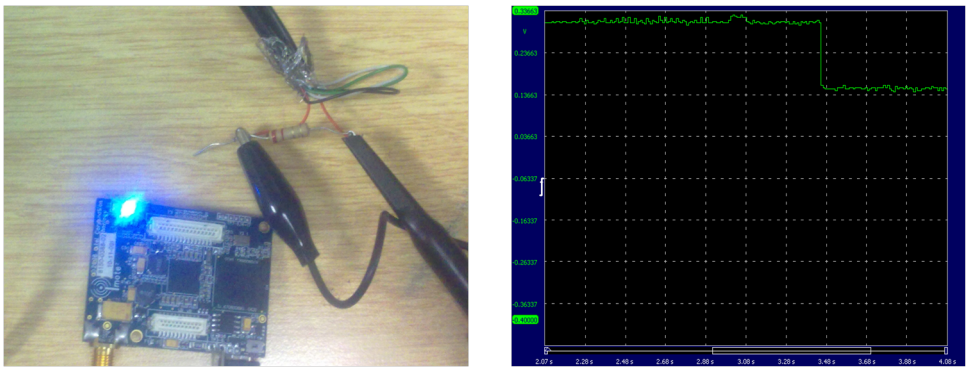

Figure 4.4: Set up to Measure Current

Energy is a scarce resource of WSNs and needs to be consumed intelligently. We use PC based oscilloscope for actual current measurement. Figure 4.4 shows experimental setup to measure energy. Resistance of 2.2ohms is introduced and graph shown on the right of figure 4.4 obtained are processed in Matlab to find current values.

A. Transmission/Reception Energy: Transmission and reception energy is computed using:

$$Tx/Rx\_Energy = Voltage * Current * Total\_Bits/Bit\_Rate$$

where *Voltage* are calculated with voltmeter set to $4.5V$. To transmit at 0 dBm, CC2420 draws a current of $17.4mA$ and operating at $104MhZ66mA$ current is consumed [25]. Summing up the two current values gives us the total transmission current. In reception $18.4mA$ current is drawn. Total time is computed by dividing the total number of buts transmitted by bit rate i.e. 250 kbps.

B. Computation Energy: Using Imote2, we have captured physical energy consumption for each scheme under consideration. Total energy con-

sumed by the mote for encoding data at source node and decoding at relay and destination node is calculated by the following formula:

$$Encoding\_Energy = Voltage * Current * Execution\_Time$$

where $Execution\_Time$ is time to encode the data or to decode a packet

C. Network Communication Energy: IEEE 802.15.4 specifies that there is a header of 9 bytes for every data packet [20]. If there are N hops between source and destination then total energy consumption in such network is calculated by:

$$N(Tx) + N - 1(Rx)$$

4.3.1 Packet Reception Ratio

Packet Reception Rate (PRR) is a function of Bit Error Rate (BER) and packet size, i.e.

$$PRR = (1 - BER)^f$$

where f is the packet size in bits and

$$BER = f(\frac{Eb}{N_o})$$

where (Eb/N_o) is the ratio of average energy per bit to single sided noise power spectral density. Signal to Noise Ratio (SNR) can be calculated as:

$$SNR = \frac{Eb}{N_o}.\frac{R}{B}$$

where R is the data rate and B is the bandwidth. Since SNR is the ratio of received signal strength to the channel noise power therefore SNR can roughly be approximated by using the following equation:

$$SNR = \frac{RSSI}{N}$$

or

$$RSSI = SNR.N = \frac{Eb}{N_o}\frac{R}{B}N$$

where N is channel noise power and can be calculated as:

$$N = kTRB$$

where k is Boltzman Constant $1.38x10^6$, T is temperature , R is data rate which $250kbps$. So,

$$\frac{Eb}{N_o} = \frac{RSSI}{kTR}$$

Since k, T are constants, Eb/N0 only depends upon the RSSI and data rate R.

As defined above, Bit Error Rate (BER) is a function of $\frac{Eb}{N_o}$ and PRR is again a function of BER. Hence, PRR is a function of RSSI and R. We get:

$$PRR = g.\frac{RSSI}{R}$$

4.3.2 Noise Floor Estimation

Noise floor can also be obtained by sampling RSSI register with no motes transmitting enabling us to have a rough approximation of SNR using following equation:

$$SNR \approx 10log_{10}\frac{RSSI - P_n}{P_n}$$

where Pn is the noise floor. The above equation is an estimation and neglects interference from other sources that maybe included in the $RSSI$ value. Noise floor can be analytically calculated by:

$$P_n = FkT_oB$$

where F is noise figure of CC2420 radio equal to 11/12 dB, T_o is ambient temperature , B is noise equivalent noise bandwidth approximately 3MHz. So, P_n comes out to be:

$$P_n(dB) = 12 - (-228.6) + 10log(300) + 10log(3x10^6)$$

$$P_n(dB) \approx -97dBm$$

This value obtained would be fed to decoder at receiving node to calculate the likelihood ratios in recovering codeword.

Chapter 5

Experimental Setup and Evaluation

5.1 Sensor Network Implementation

Actual sensor network implementation has been achieved via the Crossbow Imote2 platform [33]; designed around the low-power PXA271 XScale microcontroller with 802.15.4 radio and surface mount 2.4 GHz antenna. Following table shows the experimental setup.

Table 5.1: Experimental Set Up

Parameter	Value
Number of Motes	3 Crossbow Imote2
Data Rate	250 Kbps
Topology	Line
Channel	AWGN
Channel Estimator	RSSI
Payload Size	32 bytes
Transmission Power	3dBm

Data Collection

We used three Crossbow Intel Imotes to collect data and evaluate the performance of reliability protocols. The Imote2 comes with TinyOS pre-installed however we need Linux to be installed on Imote in order to evaluate FEC based reliability protocol. Since no bootloader or similar mechanism is installed to allow re?ashing of the device. JTAG available on the interface

board is used with OpenOCD and a JTAGKeyTiny adapter to bootstrap Linux system on Imote2. Details of building Linux image for Imotes and flashing images can be found in appendix A and B.

On top of Linux, we used following message structure during the communication between the motes. the RSSI value anytime from the chip register. RSSI gives us a good estimation of signal strength. For received packet, RSSI is the sum of noise level and packet signal strength. When there is no traffic, RSSI measures the ambient RF energy.

The test-bed is WiFi enabled indoor environment. The environment is under controlled. There is no moving object and sensors are places statically during experiments. Among the three motes, one is sender middle one is relay node responsible for forwarding data to receiver. The source can communicate with the receiver through relay node only. Event is generated for the sender and sender generate data bits, combine data bits with parity symbols (for LDPC encoding) and sends packet to relay nodes after appending header.

Message Structure

We have used message structure compliant to TinyOS MAC layer. Following is the message structure used for the current thesis.

Table 5.2: Fields in Message Structure

_u8length;	Data length of Payload
_u8fcfhi;	Frame Control Field higher byte
_u8fcflo;	Frame Control Field lower byte
_u8dsn;	Sequence Number
_u8destpan;	Destination PAN
_u8addr;	Destination Address
_u8type;	Type ID ; 1 for Data, 2 for ACK
_u8group;	Group id as the same as tinyos has
_s8data[TOSH_DATA_LENGTH];	Payload
_u8strength;	RSSI
_u8lqi;	LQI
_u8crc;	CRC
_u8ack;	Acknowledgement
_u16time;	Time Stamp of Packet

The type of the fields is an unsigned integer with the given amount of bits. The first field defines the data length of the payload, followed by a 2 bytes Frame Control Field and a sequence number. Addressing is divided into

the specification of the destination PAN8 and the mote ID, each represented by 2 bytes. After setting the Active Message Type as of TinyOS and a user defined group id, the payload is represented as an array with the size defined by TOSH_DATA_LENGTH. In case of the Imote2 default payload is 28 bytes long. The following fields are destined for transmission purpose such as received signal strength indicator (rssi) link quality indicator (lqi) and the cyclic redundancy check (crc).

We use distance as attenuator to adjust RF transmission energy. Since the noise is stable in the testbed, we get different SNR by collecting data at difference distance.

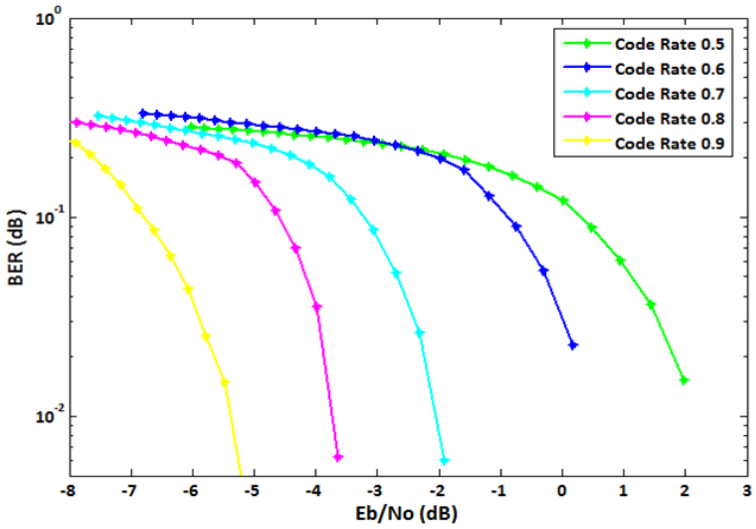

Figure 5.1: Performance of LDPC Codes over AWGN Channel

The maximum packet payload size is 116 bytes for Imote2 and by default it uses 28 bytes. We carried out experiments for packet payload size 32 bytes with varying size of data bits. The size of data bits is dependent on the code rate used for the current scenario. LDPC codes are used for error detection and correction in proposed HARQ framework. Figure 5.1 shows the performance of LDPC codes over AWGN channel against different code rates. Higher code rate shows (0.9) shows pretty good performance over Code rate 0.5 covers operates over a wide range of SNR however resource constrained sensors restricts us from using such lower code rate for transmission in fairly good channel quality.

Figure 5.2 shows performance of LDPC codes over AWGN channel in terms of iterations required to perform to retrieve the original codeword.

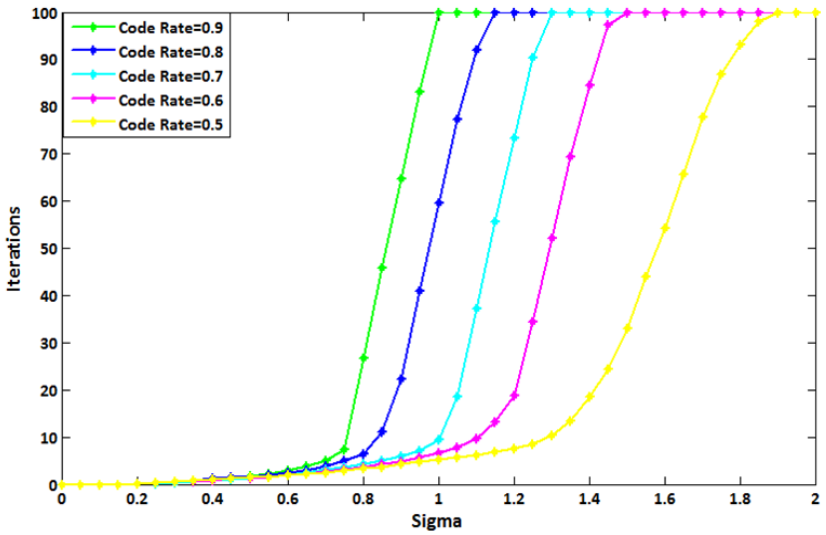

Figure 5.2: Performance of LDPC Codes in terms of Iterations

Number of iterations help in selecting the code rate according to channel quality. Higher code rate (0.9) helps in achieving satisfactory performance as far as channel quality is good while code rate of 0.7 is useful for moderate and 0.5 for worst channel conditions.

Figure 5.3 shows the performance of traditional data delivery schemes ARQ, FEC and proposes HARQ scheme in terms of Bit Error Rate against different channel conditions. In ARQ as channel condition degrades, corruption in uncoded packet increases and packets get completely corrupted when signal strength is reported with signal strength higher than -45dB. Talking about FEC, when no partial processing is performed and relay nodes perform as-is forwarding, tolerable error correction and detection can be seen till -60dB. Corruption increases as signal strength drops beyond -60dB. Significant decrease in BER can be seen with proposed HARQ scheme. With partial processing at intermediate nodes helps in achieving lower BER.

One of the main objective of this research is to reduce the number of retransmissions. ARQ, FEC with end-to-end retransmission and HARQ are compared in figure 5.4. Maximum retransmission limit has been set to 3 for all schemes to make a fair comparison. ARQ fails to deliver packet in channel condition with RSSI 0f -44/-45dB and even after maximum retries limit message received by the receiver is completely corrupted and is useless. With FEC end to end retransmission is performed and for HARQ hop by hop retransmission is done. Selection of code rate in accordance to prevailing network conditions and partial processing helps in reducing number of

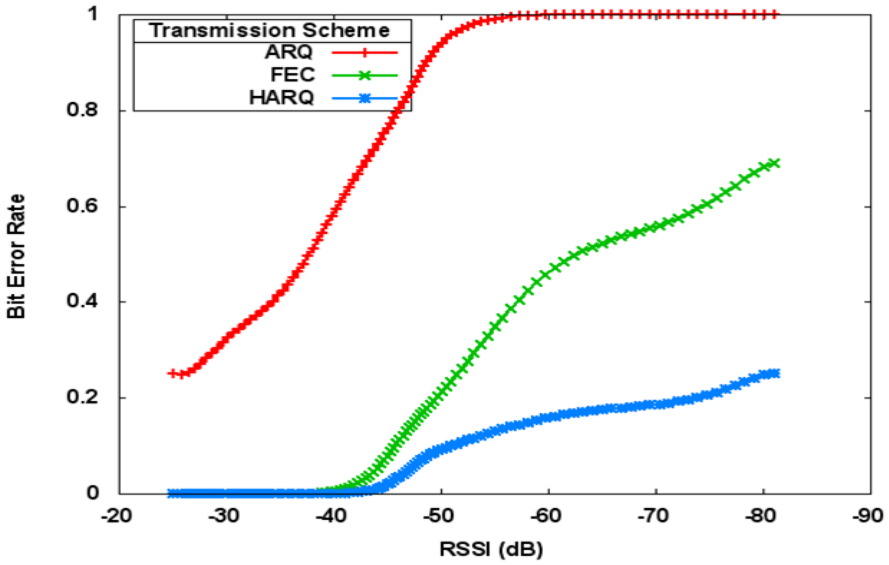

Figure 5.3: BER for Different Schemes

retransmissions.

Energy is one of the most important resource of sensor node. The current research target reliability critical application therefore we aimed to provide maximum reliability at reasonable energy consumption. Energy consumption in the network by different schemes is shown in figure 5.5

Proposed HARQ is compared against ARQ, FEC, FEC with end-to-end error recovery and FEC with partial processing but end-to-end error recovery mechanism. Considering ARQ, sender sends a packet and waits for an acknowledgement by the receiver. When channel condition is good enough, ARQ outperforms the other schemes however as channel quality starts to degrade, number of retransmissions start increasing resulting in more energy usage as compared to other schemes. This is due to the fact that energy consumed in communication is higher than local processing, In case of FEC, there is no error recovery mechanism therefore, constant energy consumption can be seen in the figure 5.5. For, FEC with E2E error correction is provided however in worst channel condition retransmission costs higher in terms of energy consumption. If partial processing is done at each relay node and end to end error recovery is provided, some decrease in energy consumption is observed however again the same coding rate for all channel conditions and end to end retransmission consumes higher energy.

Significant decrease in energy consumption of proposed HARQ can be observed. One of the major reason of this significant decrease is channel

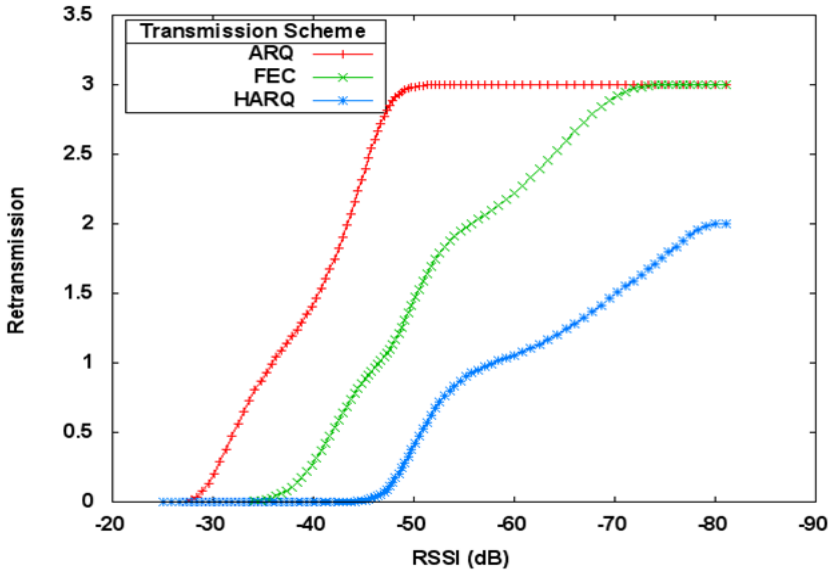

Figure 5.4: Retransmissions

awareness. Since encoding is done according to prevailing network condition and receiving node is channel aware therefore, decision of performing processing (partial decoding) is made accordingly. When channel of higher quality is available for transmission then there is no requirement for the intermediate node to process the data rather it just as-is forward the packet. When channel errors increases, intermediate nodes perform partial processing which helps in partially recovering the corrupted and achieving lower BER also evident from figure 5.3. If the packet has traversed the channel at instant when channel quality was low, hop-by-hop error recovery mechanism not only helped in cutting down the transmission cost all the way from source to receiver but also enabled the source node to perform encoding with higher code rate protecting data from future errors reducing retransmission(s).

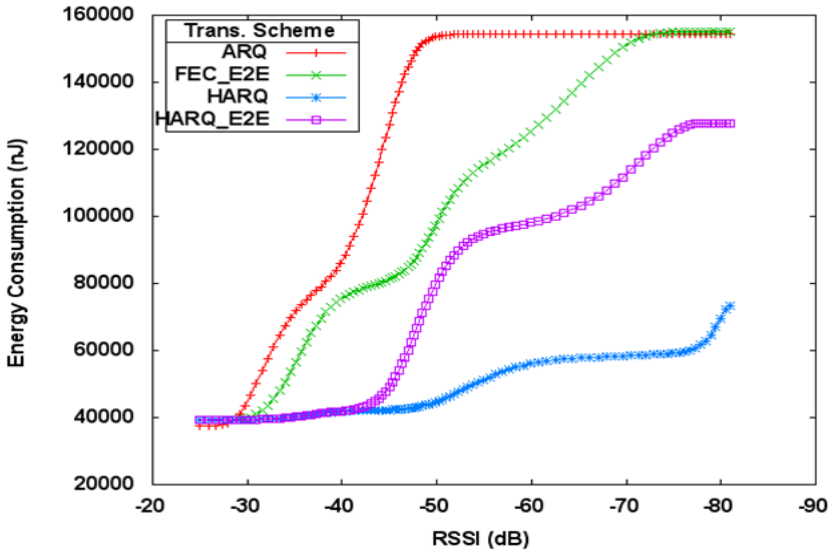

Figure 5.5: Energy Consumption in the Network for Different Reliability Providing Schemes

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Wireless Sensor Networks have proven themselves as indispensable part of our daily lives. Despite the application nature targeted by WSN, we are ultimately interested in reception of correct data collected by sensor nodes especially in event-driven networks. Due to the resource constrained nature of sensor networks, achieving reliable data communication is not a trivial task with efficient utilization of communication resources. Designing energy-efficient data communication mechanisms plays a vital role in maximizing the life-time of wireless sensor networks. Since communication is accomplished through lossy error prone wireless link where errors are introduced in a packet while transmission from the source to destination therefore it is highly required to protect our data from errors induced by the channel. Motivated by above mentioned challenges, we have proposed a Hybrid ARQ for event driven wireless sensor network where data reliability is critical. Proposed framework is adaptable to dynamic network conditions and is capable of transmitting data reliably with least amount of retransmissions. Our work is based on the fact that rate of energy depletion is higher in transmission and reception as compared to processing at sensor CPU. We integrated Low Density Parity Check (LDPC) codes in order to achieve high level of reliability in distributed fashion for efficient data communications.Partial Processing is proposed to be performed at relay nodes that helps in achieving reduced Bit error rate at destination. Iterative nature of LDPC codes helps in partially recovering the corrupted data and thus minimizes retransmission and saves energy. However partial processing is dependent on channel condition. If RSSI at receiving node reports degraded channel quality and node has sufficient energy then node is required to perform 1 iteration on packet. However

if channel quality is reported with good quality then relay nodes would just forward the data.

Framework is implemented using Crossbow Intel Imote2. Actual on-sensor implementation of the proposed and traditional techniques enabled us to show the efficacy of our framework with fair comparison. PC based oscilloscope is used to measure current and actual energy measurements are done with actual transmission, reception and processing of data and/or packet.

6.2 Future Work

We intend to test our proposed framework with large number of nodes and therefore with clustered network. Routing and reliable data transmission is very interesting extension of this problem. It also comes up with optimization problem and task assignment to different nodes in the network. Moreover, currently we performed experiments with simple data. We would like to test our HARQ framework with different types of data for instance images and/or videos.

Investigation of joint source and channel coding is another aspect of extending the current work. Another interesting research problem is an optimization problem regarding number of iterations required to be performed at relaying nodes. We have proposed a heuristic regarding the number of iterations being performed by intermediate nodes so its experimental evaluation is primarily the first step towards the extension of this thesis.

Bibliography

[1] C. Y. Wan, A. T. Campbell, and L. Krishnamurthy, "A reliable transport protocol for wireless sensor networks", *ACM International Workshop on Wireless Sensor Networks and Applications*,pp.1-11, Sept. 2002.

[2] F. Stann and J. Heidemann, "Reliable data transport in sensor networks," *IEEE International Workshop on Sensor Network Protocols and Applications*, pp.102-112, May 2003.

[30] A. Ayadi, P. Maille, D. Ros, "Improving Distributed TCP Caching for Wireless Sensor Networks", *Proceedings of the 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net 2010*, 978-1-4244-8436-2, IEEE, Juan Les Pins, France (2010), pp. 16

[4] M. C. Vuran and I. F. Akyildiz, "Error Control in Wireless Sensor Networks: A Cross Layer Analysis", *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1186-1199, Aug. 2009.

[5] N. Baccour, A. Kouba, M. Jama, H. Youssef, M. Zuniga, M. Alves, "A Comparative Simulation Study of Link Quality Estimators in Wireless Sensor Networks", *17th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'09)*, UK, September 21-23, 2009.

[6] C. Guo, J. Zhou, P. Pawelczak, and R. Hekmat, "Improving packet delivery ratio estimation for indoor ad hoc and wireless sensor networks", *Proceedings of IEEE CCNC 2009*, Las Vegas, Nevada, U.S., Jan. 2009.

[7] J. H. Kleinschmidt, W. C. Borelli, M. E. Pellenz, "An Analytical Model for Energy Efficiency of Error Control Schemes in Sensor Networks", *ICC 2007, 3895-3900*, 2007.

[8] R. Gallagar, "Low-density Parity Check Codes", *IRE Transactions Information Theory*,pp. 21-28, Jan, 1962.

[9] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity approaching irregular low-density parity-check codes", *IEEE Transactions on Information Theory* ,47(2),pp. 619-637, 2001.

[10] G. Pei, C. Chien, "Low power TDMA in Large Wireless Sensor Networks", *Proceedings of IEEE MILCOM 2001*, pp. 347- 351, Vol. 1, 2001.

[11] W. Ye, J. Heidemann, D. Estrin, "An Energy-Efficient MAC protocol for Wireless Sensor Networks", *Proceedings of IEEE Infocom, USC/Information Sciences Institute, IEEE*, New York, NY, USA, pp. 1567- 1576, 2002.

[12] S. De and H. D. Cavdar, "Channel-aware link layer ARQ strategies in Wireless Networks", *Proceedings IWCMC '08*, Crete Island, Greece, Aug. 2008.

[13] K. S. Kumar, R. Chandramouli, and K. P. Subbalakshmi, "On Stochastic Learning in Predictive Wireless ARQ", *Wireless Communication and Mobile Computing*, 8(7):871-883, 2008.

[14] M. Zorzi and R. R. Rao, "Energy Constrained Error Control for Wireless Channels", *IEEE Personal Communications Magazine*, December 4, pp. 27-33, 1997

[15] Weihuan Shu, Kumar Padmanabh, Puneet Gupta, "Prioritized Buffer Management Policy for Wireless Sensor Nodes", *International Conference on Advanced Information Networking and Applications Workshops*, pp.787-792, 2009.

[16] H. Karl , A. Willig, "Protocols and Architectures for Wireless Sensor Networks", *John Wiley & Sons*, 2005.

[17] M. Sartipi, F. Fekri, "Source and Channel Coding in Wireless Sensor Networks using LDPC Codes",*ICSACN*, pp. 309-316, October 2004.

[18] J. Zhu, S. Chen, B. Bensaou, K.-L. Hung, "Tradeoff between Lifetime and Rate allocation in Wireless Sensor Networks: A Cross Layer Approach", *Proceedings of the IEEE INFOCOM*, 2007.

[19] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks", *IEEE Transactions on Wireless Communications*, (4), 2002 660670.

[20] D. O'Rourke and C. Brennan, "A Practical Implementation of an Improved Packet Combining Scheme for Wireless Sensor Networks", *Proceedings of the ICC*, 2008.

[21] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica, "Flush: a reliable bulk transport protocol for multihop wireless networks", In SenSys 07; *Proceedings of the 5th international conference on Embedded Networked Sensor Systems*, pages 351365, Nov. 2007.

[22] S. Lin and D. Costello, "Error Control Coding", *Prentice Hall*, Second edition, 2004.

[23] T. T. Kwok, Y. Kwok, "Computation and Energy Efficient Image Processing in Wireless Sensor Networks Based on Reconfigurable Computing", *International Conference on Parallel Processing Workshops (ICPPW'06)*, 2006.

[24] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey", *Computer Networks Elsevier*, 38 (4) (2002) 393422, 2005.

[25] Mina Sartipi, F.Fekri, "Distributed source coding using short to moderate length rate-compatible LDPC codes: the entire Slepian-Wolf rate region", *IEEE Transactions on Communications 56(3)*, pp. 400-411, 2008.

[26] S.B. Qaisar, H. Radha, "Optimal Progressive error Recovery for Wireless Sensor Networks using Irregular LDPC Codes", *CISS*, Paper#80.

[27] J. Paek , R. Govindan, "RCRT: rate-controlled reliable transport for wireless sensor networks", *Proceedings of the 5th international conference on Embedded networked sensor systems*, November 06-09, 2007, Sydney, Australia.

[28] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", *RFC 3758, (Proposed Standard)*, May 2004.

[29] A. Ayadi, "Energy-Efficient and Reliable Transport Protocols for Wireless Sensor Networks: State-of-Art", *Wireless Sensor Network*, Vol. 3 No. 3, 2011, pp. 106-113.

[30] A. Dunkels, J. Alonso, T. Voigt, H. Ritter, "Distributed TCP Caching for Wireless Sensor Networks", *Proceedings of the 3rd Annual Mediterranean Ad-Hoc Networks Workshop,* Bodrum, Turkey (June 2004).

[31] B. Marchi, A. Grilo, M.Nunes, "DTSN - Distributed Transport for Sensor Networks", *Proceedings of the IEEE Symposium on Computers and Communications (ISCC 2007)*, Aveiro, Portugal, 2007.

[32] Tuan, Le, Hu, Wen, Corke, Peter, S.Jha , "ERTP : Energy-efficient and reliable transport protocol for data streaming in wireless sensor networks", *Computer Communications*, 32(7-10), pp. 1154-1171, 2009.

[33] Crossbow Inc., "Imote2 Hardware Reference Manual", September 2007. `http://memsic.com/support/documentation/wireless-sensor-networks/category/6-user-manuals.html?download=56\%3Aimote2-hardware-reference-manual`

[34] Texas Instruments, "CC2420 Datasheet", March 2007. `http://www.ti.com/lit/gpn/cc2420`

Chapter 7

OpenEmbedded

OpenEmbedded (OE)[1] is a build framework for embedded Linux and provides a collection of configurations to create Linux images. These configurations provide information about the distribution the images are build from and the machine type they are supposed to run on. The actual building is done by BitBake; task-based build tool written in Python.

Both BitBake and OpenEmbedded are constantly changing. In case of BitBake the latest version should be obtained from the project page[2]. As explained below OpenEmbedded is installed from a git repository. Therefore, it is a development snapshot which is updated on regular basis.

7.1 Installation

OpenEmbedded is currently supported on UNIX-like operating systems only. Number of dependencies are required to build OE successfully and lists of all required software and tools is provided at[3]. Installation instructions for all supported operating systems are provided at[4]

Following details are based on "Getting Started" instructions on project webpage[5].

7.1.1 Creating Directory

Base directory of your Openembedded environment (mostly /stuff/) is the location where sources will be checked out (or unpacked). To create a direc-

[1] `http://www.openembedded.org/`
[2] `http://developer.berlios.de/projects/bitbake/`
[3] `http://www.openembedded.org/wiki/Required_software`
[4] `http://wiki.openembedded.org/index.php/OEandYourDistro`
[5] `http://www.openembedded.org/wiki/Getting_started`

tory structure:

```
$ mkdir -p /stuff/build/conf
$ cd /stuff/
```

7.1.2 Obtaining BitBake

Latest version of BitBake is mostly not included in the host repositories, therefore it should be installed from the BitBake installation page

`http://developer.berlios.de/projects/bitbake/`

It is recommended to run bitbake without installing it, as a sibling directory of openembedded/ and build/ directories. Unpack the downloaded archive and modify 'PATH' variable. Modification of 'PATH' variable completes installation. Temporary modification is done using following command:

```
export $PATH= /some_directory/bitbake/bin:$PATH
```

This modification can be made permanent by appending the mentioned command to "/home/user_name/bash_profile"

7.1.3 Obtaining OpenEmbedded

Openembedded resides in a Git repository. In order to obtain OE we need to:

1. Install git
2. Go to base directory of OpenEmbedded environment

Following command is used to check OpenEmbedded from a git repository which will create a directory "openembedded" and download all contents into it:

```
$ cd /stuff/
$ git clone git://git.openembedded/openembedded
```

Updating

It is a good practice to update OE frequently. To do so execute the following command:

```
$ git pull −−rebase
```

7.1.4 Local Configurations

Important locations of open embedded are the directories recipes, sources and build. Recipes directory contain BitBake configuration files which are part of OE and define build process and required dependencies. BitBake download the required packages into the source directory and put the build output into the location defined by "build".

We can use default local.conf.sample file and modify it to according to the requirements of host system. In order to use the default file, copy the file from 'openembedded' folder to 'build' folder as:

```
$ cd /stuff/
$ cp openembedded/conf/local.conf.sample build/conf/local.conf
$ gedit build/conf/local.conf
```

Since kernel modules would be build into the image therefore we need to customize both Imote2 machine configuration and the imote2-image configuration as described below.

Imote2 Machine Configuration

For machine configurations, we need to make changes to '/build/conf/local.conf', '/openembedded/conf/machine/imote2.conf' and '/openembedded/recipes/images/imote2-image.bb'.

In */build/conf/local.conf*, *BBFILES* specifies the location of OE configuration files for BitBake while *MACHINE* sets the target platform. *DISTRO* variable specifies the used distribution and can theoretically be set to one of the distributions mentioned in /stuff/openembedded/conf/distro. However best choice for Imote2 platform is the 'Angstrom' distribution.

Following changes would be made to **/build/conf/local.conf**file.

```
# local.conf
# location where BitBake should place the downloaded sources into
DL_DIR = "/stuff/sources"

# location of the .bb files
BBFILES := "/path/to/openembedded/recipes/*/*.bb"

DISTRO = " angstrom -2008.1"

MACHINE = "imote2"

ENABLE_BINARY_LOCALE_GENERATION = "1"

GLIBC_GENERATE_LOCALES = "en_GB.UTF-8 en_US.UTF-8"

IMAGE_KEEPROOTFS = "1"

PARALLEL_MAKE = "-j 8"

IMAGE_FSTYPES = "jffs2 tar.bz2"

PREFERRED_PROVIDER_virtual/kernel = "linux"

PREFERRED_VERSION_linux-libc-headers = "2.6.33"

PREFERRED_VERSION_udev = "142"
```

Following are the machine configurations that need to be made in **/openembedded/conf/machine/imote2.conf**

```
#@TYPE: Machine
#@Name: Crossbow iMote2

#@DESCRIPTION: Machine configuration for Crossbow iMote 2

TARGET_ARCH = "arm"

PREFERRED_PROVIDER_virtual/kernel = "linux"

PACKAGE_EXTRA_ARCHS = "iwmmxt"

KERNEL_IMAGETYPE = "zImage"

IMAGE_FSTYPE += "jffs2"

EXTRA_IMAGECMD_jffs2 ="--l --pad=0x01DC0000 --eraseblock
= 0x20000"

CMDLINE="root=/dev/mtdblock2 rootfstype=jffs2 console=ttyS2,
115200"

SERIAL_CONSOLE = "115200 ttyS2"

require conf/machine/include/tune-xscale.inc

ROOT_FLASH_SIZE = "30"

MACHINE_FEATURES = "kernel26 usbgadget alsa iwmmxt"
```

Also, we need to change in the recipe file of imote2 i.e. in **/openembedded/recipes/images/image.bb**

```
# imote2-image.bb

DISTRO_SSH_DAEMON = "dropbear"

IMAGE_PREPROCESS_COMMAND = "create_etc_timestamp"

IMAGE_INSTALL = "task-boot \
util-linux-ng-mount util-linux-ng-umount \
$DISTRO_SSH_DAEMON \
lowpan-tools \
ibrdtnd \
angstrom-version \
kernel-modules \
"
export IMAGE_BASENAME ="imote2-image"
IMAGE_LINGUAS = ""

inherit image
```

Setting Environment Variable

After installation of OpenEmbedded and BitBake some environment variables are required to be set as described here:

```
export OEBASE = /stuff/
export BBPATH = $OEBASE/build:$OEBASE/openembedded
export BB_ENV_EXTRAWHITE = "OEBASE"
```

7.1.5 Building Images

BitBake uses the meta-data provided by OpenEmbedded to actually build the software and the resulting firmware and kernel images.Complete firmware image is build by executing BitBake as follows:

```
/stuff/build $ bitbake linux
/stuff/build $ bitbake imote2-image
```

The images can be found in /stuff/build/tmp/deploy/glibc/images/imote2/ once build has been finished.

Chapter 8

Linux Installation and Network Setup for Imote2

Major tasks in configuring Imote2 network for encoding and data transmission are:

- Flashing Linux kernel
- Radio Configuration
- Cross Compiler
- Assigning Node IDs

8.1 Linux Installation

Linux one of the most prominent computer operating systems is installed on a wide variety of computer hardware, ranging from embedded devices, mobile phones to supercomputers, making it suitable candidate for installation on Imote2 with powerful functionality. Linux Installation on Imote2 requires:

1. PC with Linux OS
2. OpenOCD Linux
3. Intel JTAG cable
4. IIB2400 debugger board
5. USB cable

FTDI Drivers

To work properly with Imote2 debug board and optional the Olimex ARM-USB-TINY adapter the host system needs a ft2232 USB driver. Necessary drivers for Microsoft Windows can be found on FTDI-website [1].

Virtual COM port (VCP) drivers cause the USB device to appear as an additional COM port available to the PC. Application software can access the USB device in the same way as it would access a standard COM port. A detailed installation instruction for Windows 98/ME/2000/XP can be found on the intel-research site [2]

8.1.1 Flashing images on Imotes

With a transfer rate of about 8 kB/s the Olimex ARM-USB-TINY adapter allows faster flashing of images on IMote2 .For flashing the images Open On-Chip Debugger (OpenOCD) is used.

OpenOCD on Linux

OpenOCD can be found on project page[3] and can also be obtained from git repository[4]. Latest version should be preferred always regardless of stability.

```
#bootstrap only necessary for the git-version
user@ubuntu:/path/to/OpenOCD/$ ./bootstrap
user@ubuntu:/path/to/OpenOCD/$ ./configure−−enable−ft2232_libftdi
user@ubuntu:/path/to/OpenOCD/$ make
user@ubuntu:/path/to/OpenOCD/$ make install
```

Once connection is established, next step is of flashing images on imote2 using telnet console. Following set of instructions are required to be followed for flashing:

[1] http://www.ftdichip.com/FTDrivers.htm

[2] http://embedded.seattle.intel-research.net/wiki/index.php?title=SettinguptheFTDIUSBdriver

[3] http://developer.berlios.de/projects/openocd/

[4] git://openocd.git.sourceforge.net/gitroot/openocd/openocd

```
user@ubuntu:/OpenOCD_Directory/$ telnet localhost 4444
Trying 127.0.0.1...
Connected to localhost.
Open On-Chip Debugger

#reset and halt mote
> reset halt

#disable flash protection for sectors 0 to 258 on flash bank 0
> flash protect 0 0 258 off

#erase flash
> flash erase_sector 0 0 258

#write bootloader
> flash write_image blob 0x0

#write kernel
> flash write_image zImage 0x40000

#write file-system
> flash write_image fs.jffs2 0x240000
```

OpenOCD needs to be started from the directory which contains the images and without modifying the device permissions. After starting OpenOCD a telnet connection into the debugger is established. Since to OpenOCD mote is in an unknown state it needs to be reset first. After halting the mote and disabling the flash protection, the flash memory can be erased and the images can be downloaded. Following shows how to establish connection of OpenOCD with imote2 via the telnet console.

```
user@ubuntu:/path/to/images/$ sudo openocd -f ./tcl/interface/olimex-jtag-tiny.cfg -f ./tcl/board/crossbow_tech_imote2.cfg

Open On-Chip Debugger 0.4.0 (2010-03-24-21:47)
Licensed under GNU GPL v2. For bug reports, read http://openocd.berlios.de/doc/doxygen/bugs.html

jtag_nsrst_delay: 260

jtag_ntrst_delay: 250

Info : imote2.cpu: hardware has 2 breakpoints and 2 watchpoints

jtag_nsrst_delay: 800

trst_and_srst separate srst_gates_jtag trst_push_pull srst_open_drain

Info : clock speed 6000 kHz

Info : JTAG tap: imote2.cpu tap/device found: 0x79265013 (mfg: 0x009, part:0x9265, ver: 0x7)

Info : accepting 'telnet' connection from 0
```

Configurations

Now Linux has been built on Imote2 and Imote2 can communicate with Linux computers, therefore in this section we will setup some necessary configurations to make Imote2 communicate with host PC as an 'Ethernet Gadget'.

8.1.2 USBnet Setup

After flashing images on imote, remove JTAG cable. Now connect USB to an interface board and plug into PC. After getting connected, a new interface *USB0* will appear on host PC. TO assign an IP to mote, we would configuration Imote2 IP tables with host PC as its gateway. Static IP address 192.168.99.101 is assigned to Imote while Host PC will have an IP 192.168.99.100. In order to do so, following set of instructions are required to be followed:

1). **Static IP Address on Host PC** On host PC, issue following command on terminal:

```
sudo gedit /etc/network/interfaces
```

and make changes according to following.

```
auto usb0
iface usb0
inet static address 192.168.99.100
netmask 255.255.255.0
gateway 192.168.99.0
```

2). **Restart NIC** Network Interface Card (NIC) is required to be reboot once. In order to do so issue the following command on terminal:

```
root@localhost# /etc/init.d/networking restart
root@localhost# ifconfig
```

Following configuration would be shown once the setup on host PC is successful.

```
usb0
link encapsulation: Ethernet hardware address 32:D2:83:A1:D6:48
inet address: 192.168.99.100 broadcast: 192.168.99.255 maks: 255.255.255.0
inet6 address: fe80::30d2::8ff3::fea1:d648/64 Scope: Link UP BROADCAST RUNNING MULTICAST MTU: 1500 jump number: 1
packets received: 724 error: 0 lost: 0 overflow: 0 frame: 0
packets send: 478 error: 0 lost: 0 overflow: 0 carrierwave: 0
collision: 0 send queue length: 1000
byte receive: 62960 (61.4KB) byte send: 132596 (129.4KB)
```

On terminal; issue the following command to enter Imote2:

```
root@localhost# minicom
SG2-3 login: root
Password: (No initial password. Set it as e.g. 123 by command "passwd")
root@192.168.99.101 /root #
```

Modify */etc./network/interfaces* with a static IP address and host PC as its gateway as:

```
auto lo iface lo inet loopback
auto usb0
iface usb0
inet static address 192.168.99.100
netmask 255.255.255.0
gateway 192.168.99.100
```

Run the configuration and reboot Imote2.

```
root@192.168.99.101 /root # ./configure
root@192.168.99.101 /root # reboot
```

In contrary to imote2-linux images, OpenEmbedded images boot up into runlevel 5. On current images all links to the start scripts reside in /etc/rcS.d/ and are supposed to be executed whenever the runlevel changes. This attempt failed and the Imote2 will not be reachable via the network after booting. To enable network access the respective link "S40networking" is required in /etc/rc5.d/. Connecting to the mote via the debug device as explained above allows fxing this issue with the following command:

```
# cp /etc/rcS.d/S40networking /etc/rc5.d/
```

The parameters of the dropbear SSH server are set in its start script in /etc/init.d. The designated way of changing the configuration is to create a configuration file instead of editing the script. The configuration file is named "dropbear" and is located in /etc/default/.

8.1.3 SSH Configuration

Secure SHell (SSH) allows data exchange through a secure channel between networked devices, After following configurations, we are able to communicate with Imote2 without interface board any more.

```
root@192.1689.99.101:/root # /usr/bin/ssh-keygen-f/etc/ssh/ssh-host-
rsa-key-t rsa
Generating public/private rsa key pair
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh-host-rsa-key.
Your public key has been saved in /etc/ssh/ssh-host-rsa-key.pub.
The key fingerprint is:
ba:46:aa:f6:13:41:b1:5b:7e:fa:0d:f6:53:e3:d5:fc
root@192.168.99.101 / root # /usr/bin/ssh-keygen-f/etc/ssh/ssh-host-
rsa-key-t dsa
```

8.1.4 Updating CC2420 Driver

If OE images are used then we do not need to update CC2420 driver used for wireless communication. However, if linux-images for imote2 are used then we need to update the said driver.

1. Download the driver available at `http://code.google.com/p/imote2-localization/`. Copy driver from host PC to imote2 using "scp" with command format: 'scp [file_name] [Imote IP Address]: [destination folder]'.

   ```
   root@ubuntu # cp /home/user/downloads/tos_mac.ko
   root@192.168.99.101:/lib/modules/2.6.14_r1.1/kernel/drivers/tosmac
   ```

2. Edit the */etc/modules* file and add tosmac at the end of file.

3. Restart the Imote2.

4. Issue following commands at Imote2 terminal:

   ```
   $ cat /proc/devices
   $ mknod /dev/tosmac c 240 0
   ```

Following above steps would update mote for wireless communication.