Reliable Data Transmission and Energy Efficient Sleep Scheduling for Multi- source Multi-sink Wireless Senor Networks

A Thesis Submitted to the School of

Electrical and Computer Engineering

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

In Computer Engineering

Addis Ababa Institute of Technology

Addis Ababa, Ethiopia

By

Kassahun Tamir

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

COMPUTER ENGINEERING STREAM

RELIABLE DATA TRANSMISSION AND ENERGY EFFICIENT SLEEP SCHEDULING FOR MULTI-SOURCE MULTI-SINK WIRELESS SENSOR NETWORKS

By

Kassahun Tamir

Advisor: Menore Tekeba

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL ENGINEERING

RELIABLE DATA TRANSMISSION AND ENERGY EFFICIENT SLEEP SCHEDULING FOR MULT-SOURCE MULTI-SINK WIRELESS SENSOR NETWORKS

By

Kassahun Tamir

APROVAL BY BOARD EXAMINERS

Prof. Kwon Ho Yeol ____________________

Head of the Dept. of Electrical Signature

And computer engineering

Menore Tekeba ____________________

Advisor Signature

____________________ ____________________

External Examiner Signature

____________________ ____________________

Internal Examiner Signature

Declaration

I, the undersigned, declare that this thesis work, to the best of my knowledge and belief, is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been fully acknowledged.

NAME: Kassahun Tamir Signature: ___________________

Place: Addis Ababa

Date of submission:

This thesis has been submitted for examination with my approval as a university advisor.

Menore Tekeba Signature: _____________________

Advisor Name

ABSTRACT

Multi-source multi-sink wireless sensor networks (WSNs) have got variety of application in areas that need to detect multiple environmental monitored parameters using a single sensor field/network. This type of WSN, beside its limited resources such as energy limitation like any WSNs are, has unique characteristics for instance network congestion, since multiple sensor nodes can send data to multiple/single sink at the same time.

Research works on WSNs are not matured and well-developed. There are a number of research issues that are not yet addressed. In particular reliable data transmission and energy efficiency of WSNs are among those that are the decisive and unsolved ones as much as required. So energy efficiency is increased for multi-source multi-sink type WSNs without affecting a 100% reliable data transmission.

In this thesis work, reliable data transmission is ensured by employing hop-by-hop (in every hop) loss detection and recovery. Much energy lost due to idle listening is solved using pre-scheduling i.e. sleep and active periods of nodes are pre-scheduled. These ideas are implemented using Network Simulator (NS-2) simulator. The simulation and experimental results show that the new protocols perform well under various conditions and protocol parameter settings.

This paper provides 100% reliable data transmission and reduces the energy nearly by 50% that can be wasted by IEEE 802.11. Reliability is assured by employing hop-by-hop loss detection and recovery using a hybrid of NACK and ACK-based approach. Since NACK-based scheme cannot handle the unique case where all packets in a communication are lost, we used a last single ACK feedback to make the sender sure that all packets are received successfully. We can save more energy by employing SMAC; however, the network performance such as throughput and latency of SMAC is very poor. But in this thesis work energy saving is next to reliability and network performance.

Keywords: energy efficient sleep scheduling, idle listening, multi-source multi-sink wireless sensor networks, reliable data transmission, wireless sensor networks

ACKNOWLEDGEMENT

Behind any successful work, there are more than just the author's efforts. It would not have been possible to finish this thesis work without the help of friends and kind people around me, but to only some of whom it is possible to give particular mention here.

Above all I thank the almighty God and his mother saint Mary for letting me finish this paper successfully. They kept me healthy and gave me the strength to withstand many obstacles and the painstaking effort exerted on this thesis would not have been successful without their help.

This thesis would not have been possible without the help, guidance, support and patience of my principal supervisor, Menore Tekeba, starting from reinforcing the research idea.

I am grateful to all my friends and other people who have been around me for encouraging to successfully finish this thesis. Special thanks go to my friends Lijaddis Getnet, Helen Arefine, Tewodros Endale and Simegnew Yihunie who have been encouraging me and giving comments. Without them everything would be impossible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONIMS

ACK	Acknowledgement
ARP	Address Resolution Protocol
CBR	Constant Bit Rate
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear To Send
FIFO	First In First Out
LL	Link Layer
MAC	Medium Access Control
NACK	Negative Acknowledgement
NAM	Network Animator
NS2	Network Simulator 2
OTcl	Object Oriented Tcl
ReInForM	Reliable Information Forwarding Using Multiple Paths
RTS	Request To Send
SMAC	Sensor Medium Access Control
Tcl	Tool Command Language
TCP	Transmission Control Protocol
Tk	ToolKit
UDP	User Datagram Protocol

VCS	Virtual Carrier Sense
WSN	Wireless Sensor Network

Chapter One

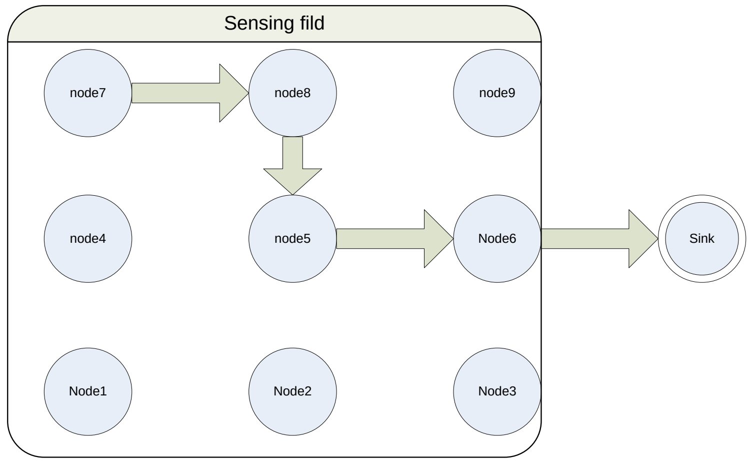
1. Introduction

A WSN consists of a group of self-organizing, lightweight sensor nodes that are used to cooperatively monitor physical or environmental conditions. Commonly monitored parameters include temperature, sound, humidity, vibration, pressure, motion and so on. Although WSN research was initially motivated by military applications, WSNs are now used in many industrial and public service areas including traffic monitoring, weather conditions monitoring, video surveillance, industrial automation and healthcare applications [1]. A basic sensor node includes five main components: an embedded processor, memory, radio transceiver, senor(s), and a power source [2]. Because of the size and cost constraints on sensor nodes, they are limited by energy, bandwidth, memory and other resources. To overcome problems that could come due to these limitations of sensor nodes appropriate design of protocol is needed. This thesis uses hop-by-hop NACK/ACK-based reliable data delivery protocol and optimized energy efficient sleep scheduling which is called pre-scheduling algorithm in multi-source multi-destination scheme WSNs to avoid reliability and energy constraints. Pre-scheduling works only for grid topology. The rest of this section is organized as follows: section 1.1 is an introduction about multi-source multi-destination scheme. Section 1.2 is all about reliable data delivery in the specified wireless sensor network type i.e. multi-source multi-destination WSN. In section 1.3, using energy efficiently in sensor nodes further more in sensor networks is discussed. The motivation and the objective of this paper are discussed in Section 1.4 and section 1.5 respectively. In section 1.6, the contributions of this research work for its user are presented clearly. The scope of the thesis is described in section 1.7. Finally this thesis organization is written in section 1.8.

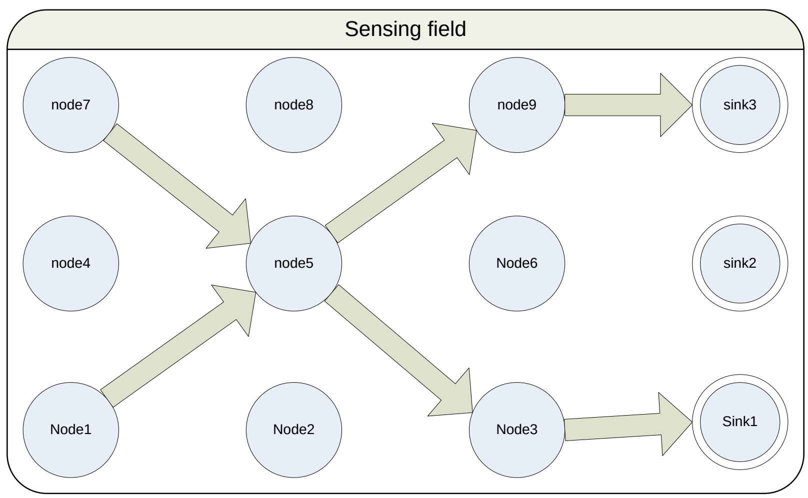
1.1. Multi-Source Multi-Destination Scheme

In WSNs, sensor nodes sense phenomenon or phenomena and process it. After processing the phenomenon or phenomena that they (multiple sources or sensor nodes) sense from the environment, they have to send the processed data to the sink or sinks. Sinks are base stations at which all the sensed data are collected that have been sent from each sensor nodes. Unlike sensor nodes sinks are not resource limited such as energy and buffer so they can be active all the time.

We can categorize WSNs according to the number of sources and sinks they have. If a WSN has only one sink and many source nodes that can transmit at the same time, we call it multi-source single-destination wireless sensor network. But in such types of networks if we restrict the network in such a way that one source node can send at a time, we call such type of networks single-source single-sink WSNs as shown in the first figure of Figure1.1a. On the other hand, if it has multiple sinks with multiple sensor nodes (sources), we call it multi-source multi-sink wireless sensor network as shown in Figure1.1b. This network architecture is obviously required when the same WSN is serving multiple applications [3], each running on distinct devices. However, the need for multiple sinks arises also in other situations. For instance, researchers are increasingly investigating the use of actuator nodes in WSNs [26]. Different actuators (sinks) take decision after they analyze or process the information gathered by sensors from the physical world and then perform appropriate actions upon the environment, which allows a user to effectively sense and act from a distance [4]. Moreover, multiple sinks are required when using distributed in-network data mining in order to discover frequent event patterns rather than transmitting raw data streams [5].As can be seen from the diagram below the second one (multi-source multi-destination scheme of WSN) is complex. Since this is the case data collision and data lose while transmitting is likely to occur, this means such networks have reliability problem, and the follow-on data recovery expends much energy of sensor nodes, energy inefficient, unless appropriate protocol is designed to overcome such limitations as much as possible. So designing reliable and energy efficient protocol for such WSNs is crucial.



a)



b)

Figure1.2- Classification of wireless sensor networks based on number of sinks a) with single sink b) with three sinks

1.2. Reliable Data Delivery

Reliable data transmission is an important issue not only in WSNs but also it is in wireless communication as a whole. Having constrained or insufficient resources in WSNs makes this very critical and decisive issue. Reliability can be defined as the number of unique data packets received by the receiver divided by the number of data packets queued to be transmitted by the sender. There exist many applications in WSNs requiring all data to be transmitted without loss. For example, structure monitoring needs the entire data from all measuring points to build a model and analyze it. Consider a sensor network deployed in a chemical plant to detect harmful gas. It is crucial for sensor nodes to reliably transport every sensor reading back to the sink. Other critical WSN applications such as biological monitoring, health care monitoring, and battlefield surveillance also require high end-to-end reliability [1], [6]. On the other hand, some applications may not require simple 100% guaranteed transmission of data packets [7]. This is because WSNs have very limited resources such as energy and bandwidth. While designing reliable protocol for such networks we should consider energy consumption in sensor nodes. Because, since we cannot change battery of sensor nodes easily, the life time of a network will be very short. Finally the network will not serve the intended objective.

In WSNs, there are plenty of factors that make the system unreliable. Such as node failure due to lack of energy source or malfunctioning of any part of it, as it is a microelectronic device. Moreover, since bandwidth of wireless networks is very narrow, congestion will occur while multiple sensor nodes attempt to transmit at the same time. Problems related to such deficiencies make designing reliable protocol challenging. So to make sure whether the data sent by the source to the destination delivered without any data loss and as much as possible to recover from loss, if it is found that there is a data loss, a receiver feedback and sender retransmission mechanism is usually used in WSNs. There are two commonly used receiver feedback mechanisms: the ACK-based approach and the NACK-based approach. In an ACK-based approach, the receiver positively acknowledges receipt of data packets, while in a NACK-based approach, the receiver only returns feedback to the sender if it detects a packet loss. The NACK-based approach incurs less overhead than the ACK-based approach, but a common problem for the NACK-based approach is that it cannot detect when all data packets are lost, since packet loss detection is based on observing gaps in the packet flow.

Reliable data transport in WSNs is a multifaceted problem influenced by the physical, MAC, network, and transport communication layers. Traditional TCP/UDP protocols and end-to-end error detection and recovery mechanism cannot be implemented directly in WSNs because of having insufficient resources. This problem can be overcome by using hop-by-hop error detection and recovery and by designing appropriate TCP. Some existing transport protocols that are designed for the provision of reliability in WSNs are [13], [14], [15] and [16]. Reliability can be provided by designing a reliable routing protocol in routing layer. For example, transmitting multiple copies of a single packet in different paths to the sink increase the probability of delivery of packets without loses [18]. RTS/CTS handshake, MAC layer acknowledgement and randomized slot selection are some of the reliability mechanisms provided by IEEE 802.11 in MAC layer [19]. Using judiciously chosen linear error correcting code, intermediate nodes in a wireless network can directly recover linear combinations of the packets from the observed noisy superposition of transmitted signals [20].

1.3. Energy Efficiency

Energy source or battery of sensor nodes are not replaceable because of they may be deployed in human unreachable areas. For example, sensors that are often intended to work in remote or hostile environment, such as a battlefield or desert; it is undesirable or impossible to recharge or replace the battery power of all the sensor nodes. But, it is possible to optimize energy usage of sensor nodes while maintaining reliable data transmission. In other words, it is possible to design a protocol by considering reliability and energy optimization as a primary and secondary task respectively i.e. by giving high priority for reliable data delivery.

The solution of energy optimization can be provided using different methods, such as, by using energy-efficient routing schemes, by using sleep scheduling method, by using appropriate MAC protocol and by designing energy efficient transport protocol. Energy-efficient routing means providing an algorithm that increases energy efficiency while routing sensed data from source to destination. For example, in [8] the data coming from different sources are aggregated at the intermediate nodes on the way; that reduces volume of data (eliminating redundancy) and saves transmission energy. This method of routing is called data-centric routing. In [9] a novel routing algorithm for energy optimization in WSNs to reduce communication distance among the nodes by transmitting sensed data through the shortest path from source node to the base station along the neighboring nodes is done. On the other way, it is also possible to increase energy efficiency of sensor nodes using sleep scheduling method. Since it is found that when operating in ad hoc mode, a node consumes nearly as much energy when idle as it does when transmitting or receiving, because it must still maintain the routing structure [10]. In sensor network applications where the traffic load is very light most of the time, it is, therefore, desirable to turn off the radio when a node doesn't participate in any data delivery. It is also possible make the node to have time at which it will not participate in data transmission or reception without affecting connectivity and area coverage. The other most important issue in provision of energy efficiency of WSNs is using appropriate MAC protocol. There are many MAC protocols proposed for sensor networks that each benefits for certain applications or under certain conditions based on the chosen design. It remains an open issue if flexible MAC protocol exists that supports various applications and operating environments while consuming minimal power and offering acceptable traffic characteristics [11]. Since this is the case, we are obliged to use appropriate MAC protocol which suits WSNs from existed ones. Among currently available MAC protocols IEEE 802.11 outperforms the two IEEE 802.15.4 CSMA-CA variants (i.e. slotted and unslotted), evidently due to the VCS mechanism in multi-hop WSNs [12]. In transport layer congestion control is a critical issue. If congestion is there in WSNs, there will be packet or data loss and the follow-on retransmission will cost high amount of energy. So by having appropriate congestion control scheme in transport layer, it is possible to optimize energy in WSNs.

1.4. Motivations

Even though their application areas are very wide, due to its complexity a network of multiple sources and multiple sinks was not attracting the attention of researchers. Many researches analysis are even considering simple network structures or topologies such as star, tree or linear topologies [12]. In this thesis work two main things are considered. First and at most the reliable data transmission for multi-source and multi-sink network is taken as the main objective. In this type of WSNs when multiple sensor nodes attempt to transmit at the same time, collision of packets would occur. If collision occurs, there will be a data loss. So reliable protocol that detect a data loss and that recover from them is necessary. The second one is designing an energy efficient MAC protocol or we can say that modified version of IEEE 802.11 protocol. Even though SMAC is the most energy efficient MAC protocol, its network performance is the poorest. So while designing energy efficient protocol, my consideration is next to reliability and network performance.

Generally, in this thesis work, the first goal is reliable data transmission, next to that energy efficient sleep scheduling with good network performances is the second issue.

1.5. Objective

General Objective: - The general objective of this paper is to design both reliable and energy efficient sleep scheduling protocol for highly congested networks i.e. multi-hop multi-sink type WSNs.

Specific Objectives:

- Identify the most suitable MAC protocol among the previously designed ones and increase its energy efficiency
- Design a link level reliable protocol which can withstand different worst cases.
- Design an energy efficient sleep scheduling protocol without affecting reliability and performance of a network

This all are accomplished for multi-source multi-sink (highly congested) and multi-hop WSNs.

1.6. Contributions

The main contributions of this thesis work are the following:

- Since this thesis work's main objective in the first rank is reliability, we could achieve 100% reliable data transmission by employing both NACK and ACK-based approaches in order to get the advantages of both protocols.
- IEEE 802.11 is the most widely used MAC protocol. But its main drawback is it wastes much amount of energy of wireless sensor nodes by the so-called idle listening. However, it out performs other MAC protocols in many network performance metrics. So by keeping these advantages of IEEE 802.11, we

designed an always-full path energy efficient sleep scheduling method. It is possible to save around 50% of the energy that can be wasted by IEEE 802.11.

1.7. Scope of The thesis

The reliability section of this paper works for any type of topology of WSNs even for the network of sensor nodes that are randomly scattered and with any amount of source and sink nodes in a given sensor network. However, it is not working for multi-cast transmission (when a single source node sends the same data to multiple receivers at a time).

Topology types used in WSNs are not only in grid topology. There are cases that sensor nodes are randomly scattered and form a sensor field. But the energy sleep scheduling designed in this thesis work is only considering a grid topology type of WSNs which are equally spaced from one another. Whatever the number of sensor nodes, rows and columns the network has, the distance among nodes should be equal.

1.8. Thesis Organization

In this chapter about WSNs, their use, their problems as a consequence of shortage of resources and the classification of WSNs due to the number of sources and sinks are introduced. Particularly, the details about multiple sources multiple sink type WSNs are presented. In chapter two, recent related works that have been done related to reliable data transmission and energy efficiency are discussed as much as possible. The reviewed papers or research works are selected that are more recent and closely related with this research. The design of the new protocols is presented in chapter three. The design of reliable protocol, energy efficient sleep scheduling and the very details of both protocols are described clearly. In chapter four simulation and performance evaluation results are discussed. The designed protocols are directly implemented in NS2. The simulation and performance of the system is tested by using different performance metrics and parameters. At last the conclusion and possible future work are described.

Chapter two

2. Literature review

Due to the fact that WSNs are recent technologies [20], research works on them are not matured and well-developed. There are a number of research issues that are not yet addressed. On the other hand, there are a number of research works done on WSNs. Specially, reliable data transmission and finding solution for energy cost are main issues of WSNs. Since it is routine to cover all works that are related to these issues, here are some of the researches that are recent and closely related with this thesis work. In section 2.1 issues that are related to reliable data transmission and in section 2.2 issues related to energy efficient protocols are presented respectively.

2.1. Reliable Data Transmission

Reliable data transmission can be provided using receiver feedback mechanisms: ACK-based or NACK-based approaches or by any other mechanisms. In ACK-based approach or commonly known as TCP/IP (if it is implemented in transport layer) the data receiver sends the feedback packet (ACK) if it has successfully received the data packet from the sender. When the data packet is sent to any receiver the time stamp will be set for its arrival and if the data packet sender doesn't receive the ACK packet from the receiver within a predefined period, the sender will consider the data packet to be lost during the transmission and will resend the unacknowledged packet. In the contrary, the receiver of NACK-based approach will send NACK packet to the data sender if and only if it detects data/packet loss.

In most wired and wireless networks that are full of resources such as energy and bandwidth TCP (ACK-based) approach is used. Since the receiver will send ACK for the sender to approve the successful arrival of each and every packet, it will exploit the bandwidth and energy of the nodes of a network, especially, when the data packet size is very small. So it is not advisable to use this approach for WSNs, because they usually have insufficient resource. The other drawback of this approach is the high feedback overhead problem. Specially, when the senders are sending for multiple receivers at a time, the acknowledgement that will come from each receiver will create congestion and data loss.

Generally, it is not advisable to employ traditional TCP/IP reliable protocols in WSNs because each sensor node in a WSN has very limited power, bandwidth and storage space and has to cope with a lossy wireless channel. Therefore, the reliable data transport protocols that are widely used in internet i.e. TCP are not suitable for WSNs. Even the modified TCP, hop-by-hop TCP protocol, used in [22] has the problem of traditional TCP which is used in wired and wireless networks that have not deficiencies of the aforementioned resources.

On the other hand, NACK-based approach is more effective than ACK-based, because it only generates an extra packet when data loss occurs. In NACK-based loss detection and recovery scheme, extra packet (NACK) overhead will occur unless it is carefully designed. Another typical NACK problem is the loss of all data packets. In a NACK-based scheme, the receiver can detect and report packet loss only if it is aware of the incoming packet. Thus, a NACK-based scheme cannot handle the unique case where all packets in a communication are lost.

The work done in [1] proposed a new reliable data delivery protocol for general point-to-point data delivery (uni-casting) in WSNs. The proposed protocol adopts a NACK-based hop-by-hop loss detection and recovery scheme using end-to-end sequence numbers. In order to solve the single/last packet problem in the NACK-based approach, a hybrid ACK/NACK scheme is proposed where an ACK-based approach is used as a supplement to the NACK-based approach to solve the single/last packet problem. The proposed protocol also has a new queue management scheme that gives priority to new data. By introducing the idea of a Ready-Bit and newer packet first rule in the transmission queue, nodes can detect and recover lost packets in parallel with the normal data transmission process. The performance of the new protocol is tested in a Crossbow MicaZ test bed. Their experimental results show that the new protocol performs well under various system and protocol parameter settings [1].

However, the wireless sensor network used in the above work is a single-source single-sink type network. In other words, in a given sensor network or sensor field, at a time, there is only one source which can send data only to one sink. Beside that all sensor nodes are active all the time and no energy consideration has been done in [1]. But in real world there are many application areas that employ sensor networks in which multiple sources can send to multiple sinks or to a sink at the same time.

Reliable information forwarding in wireless sensor networks can also be provided by sending multiple copies of a single packet along multiple paths. In [18] ReInForM is proposed to attain desired reliability. It uses redundant copies of a packet to increase its end-to-end probability of data delivery. The degree of redundancy introduced, is controlled using the desired reliability, the local channel error conditions, neighborhood information available at each node. Hence, it relies heavily on the existence of multiple paths from source to sink. On generating a packet, the source node determines the importance of the information it contains and decides the desired reliability for it. It also knows the local channel error. Such a multi-path approach would succeed only if there are sufficiently large number of paths exists from source to sink. Moreover, the lengths of these paths should be close to that of the shortest path. The reliability that can be achieved using multi-path forwarding depends on the expected number of such paths.

When a source node sends multiple copies of a single packet on multiple paths to a sink, there would be network congestion. Specially, when there are multiple sources and multiple sinks in a network and multiple sources are allowed to transmit to a single or multiple sinks at the same time, employing ReInForM for such type WSNs, packet collision would be uncontrolled [18].

In some network protocols like in [21], sending the sensed data to a data sink may be receiver initiated. Whenever a sink node wants to process or to take action after process or for any other reason, it will send initiation message for the source node. Flush [21] is a receiver-initiated transport protocol for moving bulk data across a multi-hop, wireless sensor network. It provides end-to-end reliability by using end-to-end acknowledgments i.e. by employing end-to-end NACK acknowledgment. In this protocol there are four main problems that are: first, since the data transmission takes place after sink node initiation, the urgent data sensed from the environment that has to be transmitted to the sink immediately, will not be transmitted until sink node initiate data transmission. Second, for a single node, only one source node can send at a time or there should be only a single path to a given sink. So, for the cases that multiple source nodes need to send for a single/multiple sink node(s), it doesn't work. Third, reliability is provided by using end-to-end acknowledgement as a result data loss recovery time is very high. At last this protocol employs only NACK-based approach, so it doesn't have the advantage of last single ACK used in [1] or in our protocol.

Reliability of the network can be provided a supplement in different layers beside that are reviewed above. For example, the collision avoidance and detection method of IEEE 802.11 MAC protocol are loss prevention tools in MAC layer. Whenever any sensor node senses something from the environment, it will not send it immediately to the next hop. Instead, it has to identify whether the next hop is sending/receiving or not. If the next hop is sending/receiving its own data or any others nodes data, the node with data will wait until that node complete transmitting or search for another node that is reachable and in the same path to the sink. Otherwise collision and data loss will occur.

2.2. Energy Efficient Protocols

WSNs use battery-operated computing and sensing devices; a network of these devices will collaborate for a common application such as physical and environmental monitoring. The energy source of wireless sensor network nodes is usually non-rechargeable battery. Due to this reason the life time of sensor nodes is depends on the life time of battery of sensor nodes. Even though there are many ways to prolong the life of WSNs such as charging the battery by solar, the most efficient and practical one is designing energy efficient protocol that would use non rechargeable battery wisely.

There are four major sources of energy waste in such networks. The first one is collision. When a transmitted packet is corrupted it has to be discarded, and the follow-on retransmissions to recover the lost data increase energy consumption. Collision increases latency as well. The second source is overhearing, i.e. a node picks up packets that are destined to other nodes. The third source is the control packet overhead. Sending and receiving control packets consumes energy too, and less useful data packets can be transmitted. The last major source of energy inefficiency is the idle listening, i.e. listening to receive possible traffic that is not sent. The last one, especially, relatively wastes much amount of energy in many sensor networks. If nothing is sensed, nodes are in idle mode for most of the time [20].

IEEE 802.11 is a popular MAC protocol for any types of networks. Studies show that it outperforms other MAC protocols in different scenarios even though it has a basic problem of energy wastage [12]. IEEE 802.11 devices consume large amounts of energy because all sensor nodes are active and are in idle listening state all the time. Measurements on existing sensor device radios show that idle listening consumes nearly the same power as receiving: idle listening wastes 50-100% of the energy required for receiving [20][23]. There are many protocols designed to reduce the energy wastage due to idle listening by employing sleep scheduling techniques [25]. In sensor network applications where the traffic load is very light most of the time, it is, therefore, desirable to turn off the radio when a node doesn't participate in any data delivery or it is also possible to make the node to have time at which it will not participate in data transmission or reception without affecting connectivity and area coverage which is called sleep scheduling. Due to the above reasons, idle listening waste much amount of energy as receiving does. Since we are trying to minimize the energy being wasted by idle listening, we concentrated to review protocols that employed sleep scheduling techniques as energy saving mechanism.

Among the designed energy efficient and scheduled MAC protocols for sensor networks, Sensor MAC (SMAC) [20] is the most energy efficient MAC protocol and it can save energy 2-6 times more than IEEE 802.11 MAC protocol. It is a sleep scheduling protocol and the aforementioned major sources of energy waste on the way are tried to be solved in here.

Even though SMAC dissipates least amount of energy than other MAC protocols, its network performance is very poor. For instance, regardless of the routing method and offered load used, its throughput does not exceed 5% of the channel capacity [12]. In addition to that the latency is increased, since a sender must wait for the receiver to wake up before it can send out data.

Since the basic problem of SMAC is poor network performance, other researchers tried to overcome this problem as much as they could. For instance providing periodic energy efficient radio sleep cycles while minimizing the end-to-end communication delays is designed in [23]. However, when this optimal sleep scheduling technique is designed the researchers considered only for tree and ring topologies. Application areas of WSNs are not limited only in these two types of topological structures.

The limitations that have been observed in the reviewed works, in both reliable data transmission and energy efficiency, are solved in this research work. In particular, our new protocol has the following characteristics:

- Multiple sources can communicate with multiple sinks at any time without waiting one another. While there is multi-communication at a time, the data loss detection and recovery is also maintained as it was (100%).
- The poor network performance that was observed in other protocols, such as SMAC, is improved by obtaining an all time full path between any arbitrary sources and sinks.
- The unnecessary energy waste of sensor nodes by listening packets that has not been sent, idle listening, is reduced using sleep scheduling technique called pre-scheduling.
- The feedback overhead and unnecessary computations are reduced by employing the advantages of two feedback based approaches.

Chapter Three

3. Protocol Design

This chapter presents the design procedures for hop-by-hop reliable data delivery and energy saving sleep scheduling and their implementation in NS2 simulator. The main contribution of this thesis work is to design reliable data delivery and sleep scheduling algorithm for multi-source multi-sink type WSNs. The design methodology of hop-by-hop reliable data delivery is discussed in section 3.1 and that of sleep scheduling is presented in section 3.2.

3.1. Hop-by-hop Reliable Data Delivery

3.1.1. Overview

To deliver reliable data from multiple sources to multiple sinks, our protocol employs hop-by-hop loss detection and recovery scheme. Hop-by-hop loss detection and recovery is achieved by negative acknowledgement (NACK) and last single positive acknowledgement (ACK). The protocol uses NACK and last single ACK to detect the loss of one or more packets and loss of the whole data (all packets) respectively and to recover the lost one(s).

3.1.2. New Packet Creation in NS2

When a new packet is needed to be created its packet type is defined in the header called *packet.h* which is found in *~ns/packet.h.* The new packet added in this research work is only the NACK packet and its packet type and its symbolic name is created. But the packet type ACK is already created previously. So only the values of the packet variables (values of its header fields) are changed in link layer as per its function. The header format of the ACK and the NACK packets, the basic header fields, and their importance in error detection and recovery are discussed in the following section.

3.1.3. Packet Format

For loss detection and recovery three types of packets are used, that are ACK, NACK and DATA packets. Data packets are the sensed data from the environment or what the sensor source node wants to send for any sink node that needs the data for further processing/action. All data packets share the same header format. The header of a data packet contains five important fields: **Packet_ID**, **ACK/NACK_Bit**, **Source_ID**, **Receiver_ID**, and **Sender_ID**. Their meaning and their use are presented below.

Packet_ID: This term is used to refer to the combination of the source and destination node IDs (all sensor nodes are assumed to have a unique ID) and the end-to-end sequence number (assumed to be unique for all packets between a particular source/destination pair). Since the Packet_ID is globally unique, it can uniquely indentify data packets in the network. In NS2 this term is commonly known as unique ID.

ACK/NACK_Bit: ACK/NACK_Bit is stored in the header of a data packet. The possible value of this term is 1 or 0 to indicate whether the sender requests the receiver to respond ACK or NACK for the sent packet respectively. If the packet which is being sent is the last one, the sender will set the value of this bit1 and for packets that are other than the last packet the value of this bit is set to 0. Whenever the receiving node receives a new packet it needs to check in order arrival of packets and read the value of ACK/NACK_Bit. If the ACK/NACK_Bit value in the header of the received data packet is 0 and if the arrival is in order, the receiver will send neither ACK nor NACK packet. On the other hand, if the ACK/NACK_Bit value in the header of the received data packet is 1 and if the arrival is in order, the receiver will send ACK packet to ensure all packets are received successfully. In both cases if the arrival is out of order, the receiver will send NACK packet with a sequence number equal to the last sequence number of the last packet successfully arrived.

Source_ID: Source node is a node at which packets are originally generated. In NS2 the value of this ID is set in common header of each data packet. When a single receiver node is receiving packets from multiple source nodes at the same time, **Source_ID** is used to check in order arrival of packets that come from the same source node, the value of Source_ID is very crucial.

Sender_ID and Receiver_ID: we used MAC address of the node transmitting and receiving a file as the Link Source Address. When acknowledgement needed to be sent for the sender of the data packets, the receiver of data packets sets the Sender_ID and Receiver_ID of feedback packets the value of its MAC address and Sender_ID of the sender of the data packets respectively.

Feedback packets i.e. ACK and NACK share the same header format. The header of these packets contains four important fields: **Packet_ID**, **PacketType**, **Receiver_ID** and **LastRecieved_ID**.

Packet_ID: The meaning of Packet_ID field of the header of feedback packets is the similar to that of Packet_ID field of data packets in that it is globally unique.

PacketType: When a node receives a packet at link layer from lower layers, it extracts the value of **PacketType** field. If the value of this field is ACK/NACK, its response will depend on the type of feedback. If the packet is ACK, the node will understand that all data is received successfully and it will prepare itself for future transmission/reception. On the other hand if the packet is NACK, the node will resend the lost packet according to the value of **LastRecieved_ID** field of NACK packet.

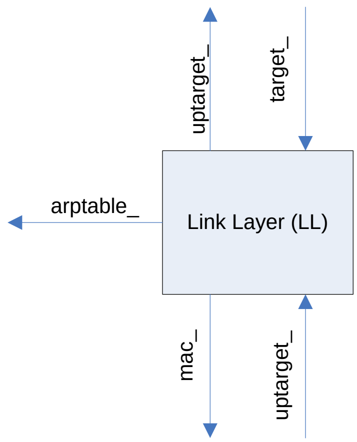
LastRecieved_ID: when the packet arrival is out of order, the receiver of data packets will set the value of **LastRecieved_ID** field in the header of NACK packet by the sequence number of the last packet successfully received. As the node receives NACK packet it would know that packets with sequence number equal and less than the value of the **LastRecieved_ID** field of NACK packet are received successfully and it would resend packets with sequence number greater than the value set in **LastRecieved_ID** field.

3.1.4. NACK and the Last Single ACK Packet Principle

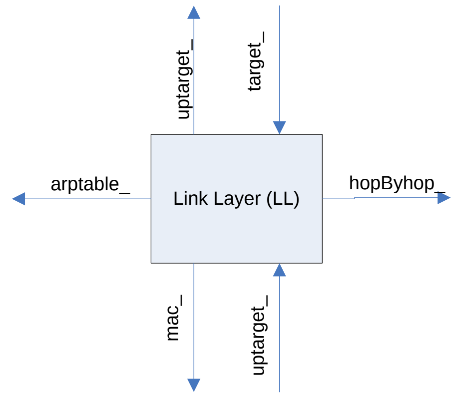
Hop-by-hop reliable data delivery is achieved by using the advantages of both NACK and ACK feedback packets. The advantage of a NACK-based approach is obvious: it is effective in detecting packet losses and also efficient in recovering from them. However, a typical problem with a NACK-based approach concerns the fact that the receiver has to be aware of the incoming packet. The advantage of an ACK-based approach is that it is effective in detecting a loss and recovering from them in that for every received packet, a TCP receiver returns an ACK packet to the sender. If an ACK packet is not received within a given timeout value, the TCP sender will assume that the packet is lost, and will retransmit the lost packet. However, feedback overhead and time delay are basic bad characteristics of ACK-based approach. Feedback overhead problem is because of acknowledging each and every data packet received. End-to-end delay is because of a sending node will not send the next packet unless ACK packet is received for previously sent packet. We took the advantages of both NACK-based and ACK-based approaches to overcome the problems in both these approaches. For all packets except the last one that would be sent from any sensor node, we used NACK-based approach. The receiver would send NACK packet if there is only a data loss. But to assure the sender of data packets as all packets are received, the last/single ACK packet is used. If the data packet is the last one, the sender would set the value ACK/NACK_Bit to 1 and sets a time stamp. If acknowledgement wouldn't be received at the sender of the data packets, it would understand that all packets are lost and would retransmit from the beginning.

3.2. Protocol Operation

The working principle of the new protocol is presented in the following sections. In this section Sender is a node from which data packets are sent and Receiver is a node to which data packets are sent. The responsibilities or the operation takes place in a sending node is discussed in section 3.2.1 and for that of receiver node is discussed in section 3.2.2.



a)



b)

Figure3.1-the architecture of link layer in mobile nodes in NS2 a) Old protocol b) New protocol

3.2.1. Old Protocol Operation

The operation of the old protocol is described based on the working principle of link layer which has been implemented in NS2 before the addition of the new protocol. The general architecture of mobile nodes and its operation is presented in Figure4.1 and section 4.2.1 respectively. Particularly in this section the general operation of link layer (LL class in NS2) is presented as can be seen in Figure3.1a.

3.2.1.1. Sender Operation

As the data packet(s) comes from the upper layer, link layer buffers the packets until their turn to be transmitted is reached and send them to the lower layer according to the default queuing principle used i.e. first in first out (FIFO). The queue length is told during nodes configuration in Tcl language as:

*$ns node-config -ifqLen $val(ifqlen) *

The life time of these packets in one's node link layer is until it will be transmitted to the lower layer.

If the packet is a routing packet, this layer will consult ARP (address resolution protocol) using a pointer *arptable_*.

3.2.1.2. Receiver Operation

When the packets, whether they are data or control packets, come from the lower layer that is MAC layer, the only responsibility of link layer is to send to the next upper layer that is network layer using a pointer *upertarget_*.

3.2.2. New Protocol Operation

Unlike the old link layer protocol, in our new protocol during both sending and receiving packets there are additional responsibilities that are detecting a loss and recover from them. The operation of detecting the loss and recovery are explained in sections 3.2.2.1 and 3.2.2.2. The functions or methods that compute the intended activities are all included in hop-by-hop.cc and hop-by-hop.h. And whenever the packet is arrived in the receiving function (*receive(p,h)*) of link layer (*LL.cc*), functions in the aforementioned files are consulted by using a pointer *hopByhop_*.

The c++ code included in the above files is explained using the flow chart (working principle error detection and recovery) shown in Figure3.2. Detecting a packet loss (if any) and recover from it is handled using the principle shown in that figure. So, whenever a single packet is received in receiving function of link layer, all the parts of the flow chart shown in the figure is executed according to their order. Its details are presented in the following section.

3.2.2.1. Sender Operation

The sender node may be a source of data to be transmitted or an intermediate/router node. Whether the sender is a source or an intermediate node, the main responsibilities of the sender node are the following.

- It copies every packet before transmitting. All packets copied in this step will be deleted when positive acknowledgement is received and when negative acknowledgement is received all packets with sequence number less than the sequence number of the lost packet will be deleted.
- If the packet being sent is other than the last packet, the sender will fill the ACK/NACK field to 0. But if the packet to be transmitted is the last one, ACK/NACK field will be set to 1.
- When transmitting the last packet, besides filling ACK/NACK field to 1, the sender would set a time stamp. If ACK packet is not received before time out, the sender will understand that all packets that are after the last negatively acknowledged packet are lost and it will retransmit the whole data packets starting from the last negatively acknowledged one again. But if there was no any negatively acknowledged packet and if ACK is not received, it will transmit the whole packets from the very beginning.
- If NACK packet is received, it extracts the **LastRecieved_ID** field and it will retransmit packets with sequence number greater than or equal to the value of **LastRecieved_ID** to the node that sends NACK packet.
- If ACK packet is received, it will understand that all data packets that are sent are successfully delivered and it will free the buffer by deleting all packets.

3.2.2.2. Receiver Operation

The receiver may receive packets in link layer from upper or lower layers. If the sender is a source node, link layer will receive packets from upper/routing layer. If this is the case, the only responsibility of a node or link layer is having a copy of and transmitting all packets. On the other hand, if packets are received in link layer from lower layers, there would be two main responsibilities of this layer. Firstly, it checks in order arrival of packets. Secondly, it sends feedback packets. Sending feedback packets depends on the result of arrival checking. If there is no loss of packets, the receiving node will send ACK packet when the last packet with ACK/NACK field set to 1 is received in order to assure that all packets are successfully received. And if there is loss of packets, the receiver node will prepare NACK packet with a sequence number or **LastRecieved_ID** equal to a sequence number of the last successfully received packet so that the sender of data packets will understand what packet(s) is to be resent.

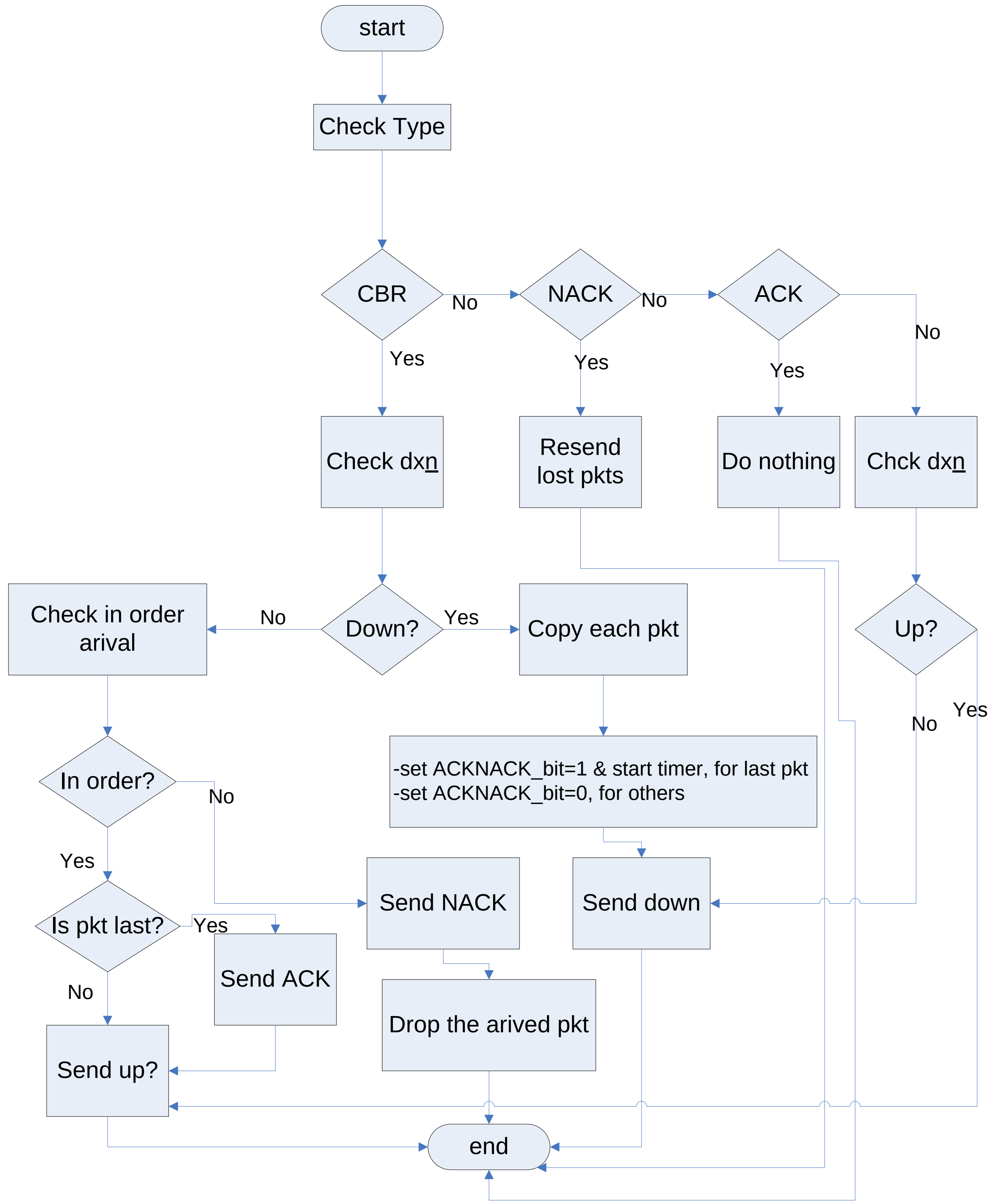


Figure3. 2 Flow chart of reliable data transmission protocol

3.3. Sleep Scheduling

3.3.1. Overview

Battery of a sensor node can be spent due to three main reasons. Over hearing, collision and the follow-on retransmission, and idle listening are the main factors that let the battery of a sensor node to die within a short period of time. Among them much amount of energy is lost due to idle listening. Therefore, the proposed design is to minimize the energy spent due to idle listening of sensor nodes. With this scheduling method, we can save around 50% of the sensor node battery power. In other words, this design can increase the life time of a sensor field or wireless sensor network by two times than a network with all sensor nodes are active all the time. Sensor MAC (SMAC) [17] can increase the life time of a network better than this design. But in SMAC the end-to-end delay and the probability of packet lost due to buffer overflow when a packet with data to be sent is waiting a node that is in the sleep period are very high.

3.3.2. Protocol Setup and Configuration

This energy saving sleep scheduling method is designed for wireless sensor network with multiple sources and multiple sinks/destinations in grid topology as shown in Figure3.2. The distance between two adjacent nodes, horizontally and vertically, is 150 meters.

Table3. 1 protocol setup and configuration

Sensing range	107 meters
Transmission range	213 meters
Distance between adjacent nodes vertically and horizontally	150 meters

We can make nodes closer or farther than the aforementioned value. But whatever the distance between two adjacent nodes is, it is assumed that it should be the same. The protocol setup and configuration are summarized in Table 3.1 and Figure3.2 below.

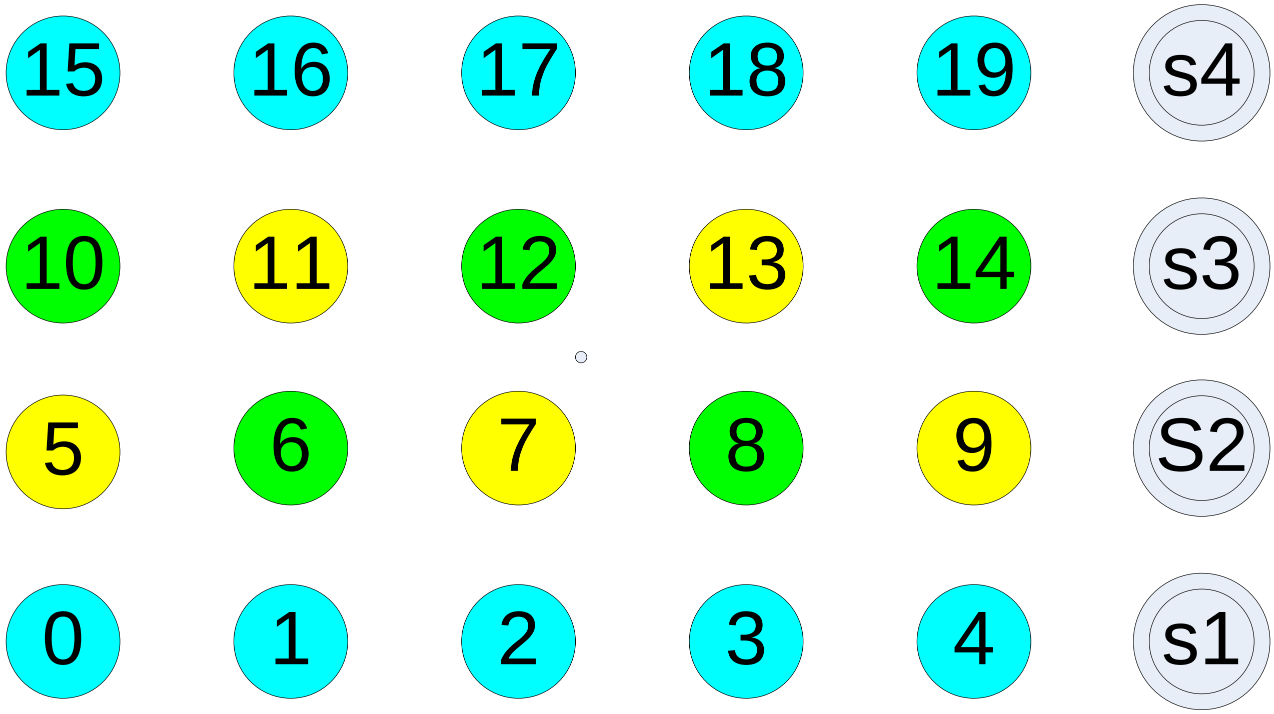


Figure3.3- The sensor field (grid topology)

3.3.3. Working Principle

In Figure3.2 the nodes painted in green are assumed to be active at one time and symmetrically nodes that are painted yellow go to sleep. And nodes that are painted in blue are active with green nodes for half time interval (1/2T or quarter cycle) and for the next half active time they are active with yellow nodes as can be seen in Figure3.3.

The sensing range of nodes we used is 107 meters and the transmission range can be greater than or equal to the square root of the sum of the square of distance between adjacent nodes i.e. it comes around 213 meters as [27] did. But it should be less than the value of two times the distance between two adjacent nodes in order to minimize the energy spent transmitting for far distance node. Particularly for this simulation the minimum transmission range is 213 meters that a sensor node can find an active node which is waiting to receive the data.

All nodes (whatever the number of row and columns are there) except nodes that are found in the upper most and lower most rows follow the two symmetrical sleeping and waking cycles Figure3.3a and Figure3.3b. But nodes found in the upper most and lower most rows use the last waking and sleeping cycle (Figure3.3d).

If the time needed to transmit from the furthest node to the sink is assumed to be ½ T seconds, the active time of nodes will be a minimum of that value (Figure3.3a). And it could have been possible to use two symmetrical sleeping and waking cycles. But for nodes found in the upper most and lower most rows, there would be isolated source nodes from the sink if the node of the second hop fails. So, to make the scheduling that tolerates the node failure in a given routing path, two methods are used.

First, we used 2T cycle of which the nodes will be active for T seconds and will be asleep for another T seconds. After this is done and if the there is a node failure in any hop, a given source/intermediate node can send through another path in the second ½ T seconds because this time interval is assumed enough to complete transmission from the source to the sink. The second one is using three sleep/wake cycles and letting the upper most and lower most rows follow the third waking/sleeping cycle which starts waking ½ T after the start and then after it follows the 2T cycle mentioned above.

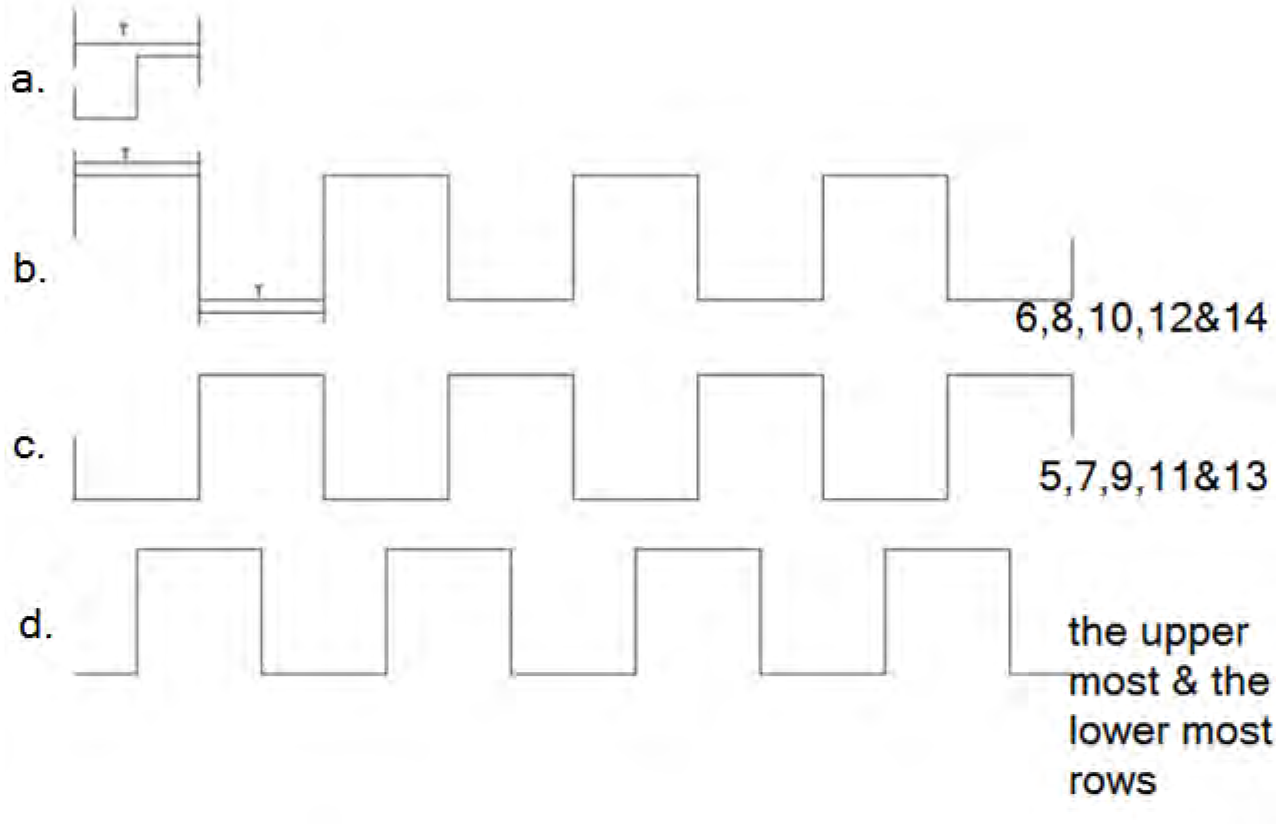


Figure3.4- The sleep/wake scheduling algorithm

Chapter Four

4. Simulation Results and Performance evaluation

This chapter provides an overview of implementation methodology, result analysis and performance evaluation of the proposed protocol. In particular, Sect. 4.1 introduces an overview of the chapter. Section 4.2 explains about Network simulator 2 (commonly known as NS2) and implementation of new protocols. Section 4.3 discusses the implementation and simulation methodology. Finally, basic tests of protocol performance are given in Sect. 4.4.

4.1. Overview

New protocols, hop-by-hop loss detection and recovery and sleep scheduling are developed in Link layer and MAC layer of NS2 simulator respectively. As described in the previous chapters the selected sensor field or network is a grid topology in such a way that multiple sensors can send to multiple sinks at the same time without waiting each other and decline in reliability of data transmission. The reliability of data transmission in this network is assured by hop-by-hop loss detection and recovery. Hop-by-hop loss detection and recovery is done by using advantages of both positive and negative acknowledgements. We used negative acknowledgement to detect and recover the lost packet(s). But if all packets are lost, a sink/next hop is unable to detect whether there is data loss or not. Even the next hop couldn't know whether data was sent to it or not. To avoid this problem, we used a single/last positive acknowledgement which should be sent for the sender after receiving all packets. Once the sender of the data receives a positive acknowledgement within a specified time limit, it can understand that all packets are received by the next hop or a sink. If not, it will resend the whole data after the last negatively acknowledged packet. In [1] this methodology is used in WSNs which have only one sender at a time. While simulating their protocol, they were also considering all sensor nodes are active all the time. But energy is one of the basic scarce resources of WSNs. There should be a sleep scheduling or another mechanism to increase life time of a sensor network. We used the former mechanism. Finally, two protocols are evaluated concurrently. In other words, we evaluated the reliability of the protocol in the presence of energy scheduling. In the next section, we presented about NS2 and the way how we developed new protocols in it.

4.2. Network Simulator 2 (NS2)

Network simulator 2 (version 2) is most commonly known as NS2. It is an event-driven simulator and it is used to simulate both wired and wireless networks. Due to its flexibility, modular nature and open source it is widely used by many researchers [24]. The internal structure/architecture of NS2 is as shown in Figure4.1. When ever needed to add special features or new modules, it is easy and is helpful. Its front end is object oriented tk/tcl language or commonly known as OTcl and its back end is c++ language. Our extension or modification is done in c++ part of the simulator.

4.2.1. New Protocol Implementation in NS2 (Simulation Methodology)

In NS2 the architecture of mobile nodes is as shown in Figure4.1. The reliable data transmission part of the new protocol is developed in link layer and that of sleep scheduling is added to the IEEE 802.11 MAC protocol. For the former one, the main responsibilities of the packet receiving function, before the addition of the new protocol, in link layer of NS2 was identifying packet types and send them according to their direction of transmission which is saved in the header of every packet.

The main responsibilities of the newly designed protocol which are described in section 3.1 are copying each packet before transmitting, detecting loss, acknowledging and resend if loss is detected. The first and the last one are the responsibilities of the sending node and the rest are of the receiving node. So whenever there is packet arrival in link layer, it will consult *hop-by-hop.cc* using *hopByhop_* pointer. All the features of the new protocol are developed in this file.

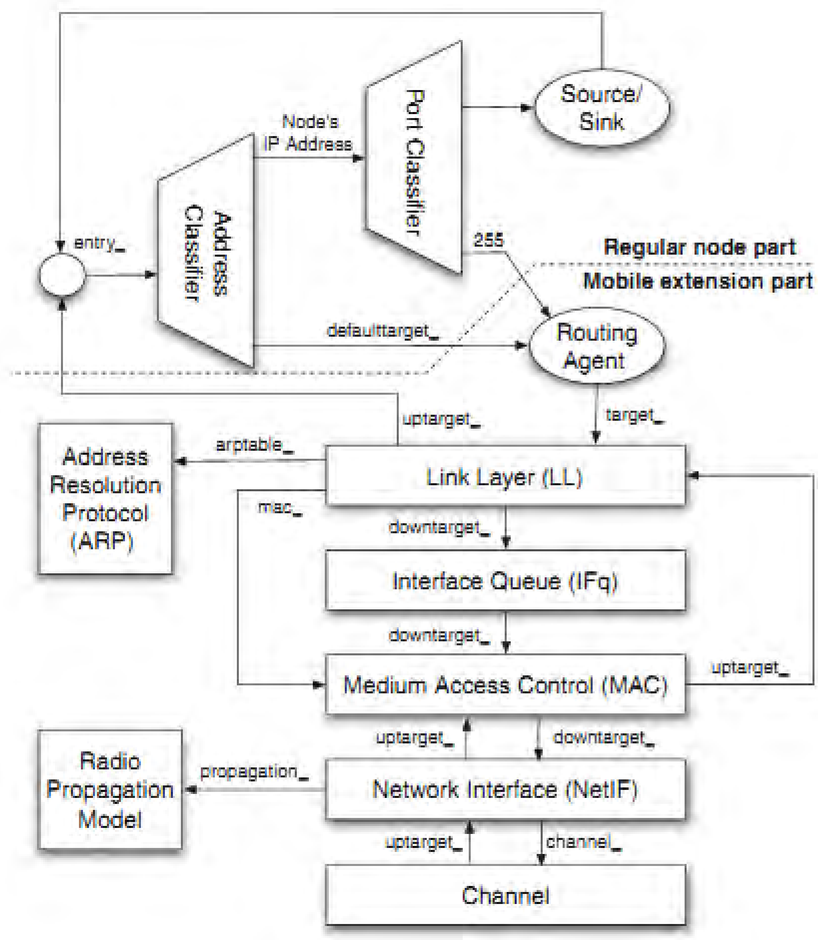


Figure4. 1- The architecture of mobile nodes. [24]

We designed the new sleep scheduling mechanism and the extension is done on IEEE 802.11 MAC protocol in that all the advantages of IEEE 802.11 are taken. The sleeping mechanism increases the life time of the wireless sensor network in almost 100% than using only IEEE 802.11. The performance of the proposed protocol (hop-by-hop error detection and recovery) is also evaluated with the sleep scheduling and its reliability/efficiency is still remains the same.

4.2.2. The OTcl Simulation Script

This is the part that the activity of the network creation, configuration and parameters settings can be done. As tried to explain in above, NS2 is written in two languages of which OTcl front end language is the one. Since the body of NS2 is fairly large, the compilation time is not negligible. A typical Pentium 4 computer requires few seconds to compile and link the codes with a small change such as including "int i=0;"in to the codes. On the other hand, any change in an OTcl file does not need compilation. C++ is fast to run but slow to change. It is suitable for running a large simulation. OTcl, on the other hand, is slow to run but fast to change. These are the reasons why NS2 uses two languages and in particular why it uses OTcl for network creation, configuration and parameter setting [24].

In this thesis work case during network creation, configuration and parameter setting, we used up to 25 nodes as per they are needed for different reasons. For example, we used many nodes when we need to increase the number of source nodes sending at a time. Other parameters are summarized in the following table and the detail is found in Appendix-A.

Table4. 1 Network configuration and parameter settings values

MAC type	Mac_802.11
Buffer Size	500
Number of Nodes	Up to 25
Transport Layer	UDP
Packet Type	CBR
Packet Size	1000 bytes
Inter Sending	0.005 ms
Number of Packets	10(for each source)
Topography Area	Mostly used 1000x1000 and 1000x1200(for scheduling)
Time of Simulation	150 ms

4.3. Evaluation Methodology

The performance of the new protocol is evaluated in this chapter from different perspectives. Specifically in this section, evaluation metrics and parameters which are used in result analysis are defined. The definitions from 1-6 are taken from [1]. In section 4.3.1 evaluation metrics are presented. At last evaluation parameters are presented in section 4.3.2.

4.3.1 Evaluation Metrics

The main objectives of this thesis work are reliable data transmission and increasing life time of a multi-source multi-sink wireless sensor network. In addition to that extrinsic values such as latency and throughput of a network are also considered.

1. *End-to-end delay*: - The end-to-end delay is measured as the interval between the generation of a data packet at its source and the reception of that packet at the sink. The average end-to-end delay for each source node is calculated as the average end-to-end delay of all data packets generated by that node. The end-to-end delay shows the average amount of time it takes for the network to deliver a data packet from a particular source node to the sink.
2. *Link Delay*: - The link delay measures the interval from when a packet is sent from the sender to the time it is received at the next hop receiver. Comparing the link delays is useful for understanding the network congestion level at each link as well as the impact of traffic load on a packet's link delay.
3. *End-to-end Reliability*: - The end-to-end reliability for each source node is defined as the number of data packets from the node that are received at the sink divided by the total number of data packets the node generates. The end-to-end reliability reflects the reliability of a given path in the network.
4. *Link Level Reliability*: - The link-level reliability measures the reliability of the link between two adjacent nodes. It is defined as the number of unique data packets received by the receiver divided by the number of data packets queued to be transmitted by the sender at every hop.

5. *End-to-end Throughput*: - The total throughput is measured as the number of unique data packets received at the sink divided by the time interval between when the first data packet is generated and the last packet is received. The achievable total throughput reflects the efficiency of the protocol. The higher the achievable total throughput, the faster source nodes can deliver their data packets to the sink. Both the end-to-end reliability and the end-to-end delay can affect the total throughput.

6. *Link level Throughput*: - The link throughput measures the throughput between two neighbor nodes. Link throughput is calculated as the number of unique data packets received at the receiver divided by the time interval between when the first data packet is generated by the sender and the last packet is received by the receiver.

7. *Total network life time*: - One of the main energy consuming factors of sensor nodes is idle listening. When compared, IEEE 802.11 is preferable than other MAC protocols in all its characteristics, except it wastes much amount of energy. To avoid this energy wastage, we used a pre-scheduling method of sleep scheduling in that all nodes are pre-scheduled in such a way that at all time there would be full transmission path from sources to sinks.

The reliability protocol of the wireless sensor network developed in this thesis work is fully efficient in both end-to-end and link level reliability measurements. The network performance regarding throughput and delay is tested: but only its end-to-end values, since, most of the time, these metrics are decisive factors and the network is intended to be the one that its end-to-end delay is small as much as possible.

4.3.2. Evaluation Parameters

The performance of WSNs, especially, in reliability case is expected to be affected by changing the following parameters. As the number of sources that can transmit at the same time increase, collision and data loss will increase proportionally. Increasing the number of hops between sinks and sources can also increase the loss of packets. In addition to that when direction of transmission is changed, particularly for X (cross) transmission direction the probability of occurrence of packet collision is very high. In our simulation, we used a random source generator and we saw the result when neighboring, one hop apart and the like nodes send at the same time. So, the performance of our protocol is evaluated by these worst cases. The parameters used are in our simulation are presented as follows.

1. *Number of sources*: - The maximum number of nodes used in our experiment is 25 arranged in a grid topology in a raw of 4 and a column of 5. To show the effect of number of sources in reliable data transmission, we used two and three source nodes in different directions of transmission i.e. when two nodes transmit in parallel and X direction.
2. *Number of hops*: - the total number of sensor nodes between a sink and source node in the path of transmission plus one is called number of hops. In our experiment from 1 to 4 hopes are used when the transmission is in parallel and maximum of 8 hops are use in X direction.
3. *Direction of Transmission (position of sources and sinks)*: - when two or more sensor nodes send the sensed data to the sink which is found in its row it is called parallel transmission. But when two source sensor nodes send the sensed data to sinks which is found in different rows and form X transmission direction while transmitting, we called it X transmission. In our experiment both of them were implemented.
4. *Random source*: - In reality, the position of multiple sources in a network is indeterminate. So to make our work realistic, we made the positions of source nodes random i.e. when two sources are adjacent or one, two or more nodes in between.

4.4. Basic Tests of Protocol Performance

The main objectives of this thesis work are design a protocol that can convey the required information from source sensor nodes to the sinks or base stations reliably and increasing the life time of sensor networks by using sleep scheduling mechanism in grid topology and in the case of multiple sources that would possibly transmit to multiple sinks or base stations at the same time as shown in Figure4.2. The designed protocol is tested from different perspectives and in all cases it is 100% reliable and can convey the intended information or data to multiple sinks without any data lose. And it is also possible to increase the life time of the network by around 100% without affecting reliability. The quality of the network i.e. throughput and delay are tested by considering different conditions beside the main objective of the network. We have seen the effect of the number of source nodes in a network i.e. scalability test in section 4.4.1. In section 4.4.2, interference and collision test is analyzed. At last in section 4.4.3 reliability and sleep scheduling together are tested.

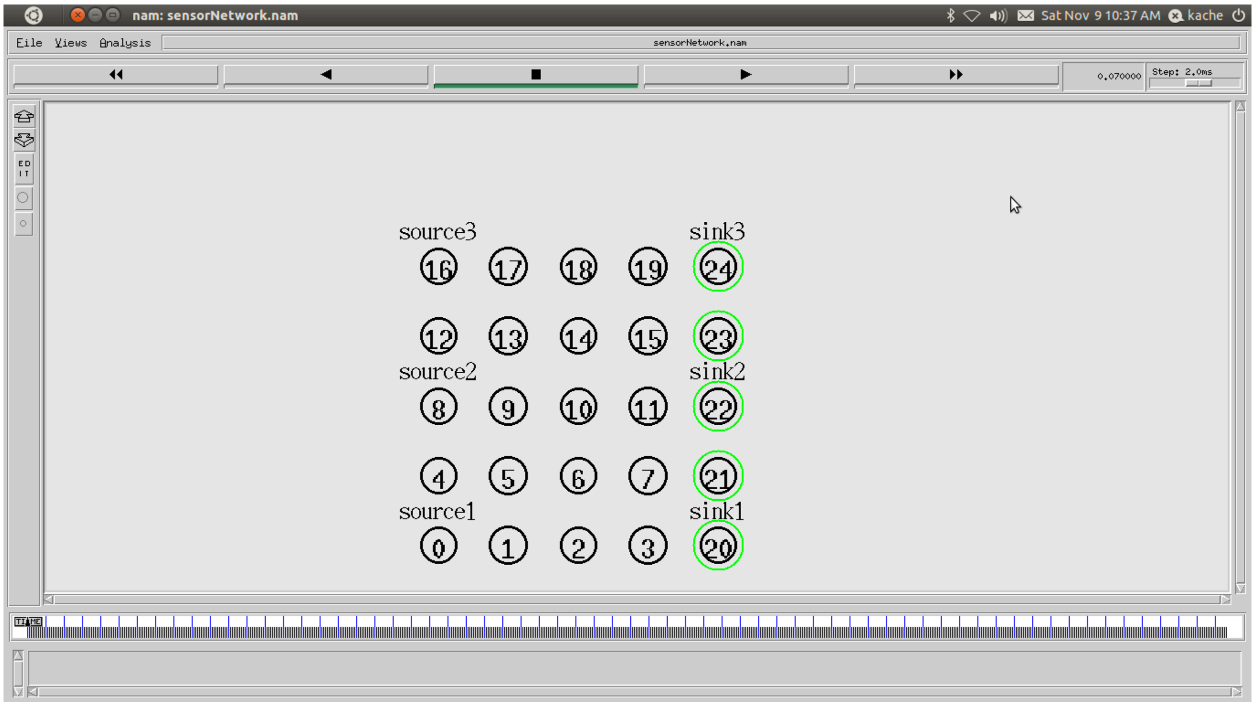


Figure4.2- Grid topology when there are three sources and three sinks that can transmit data at the same time

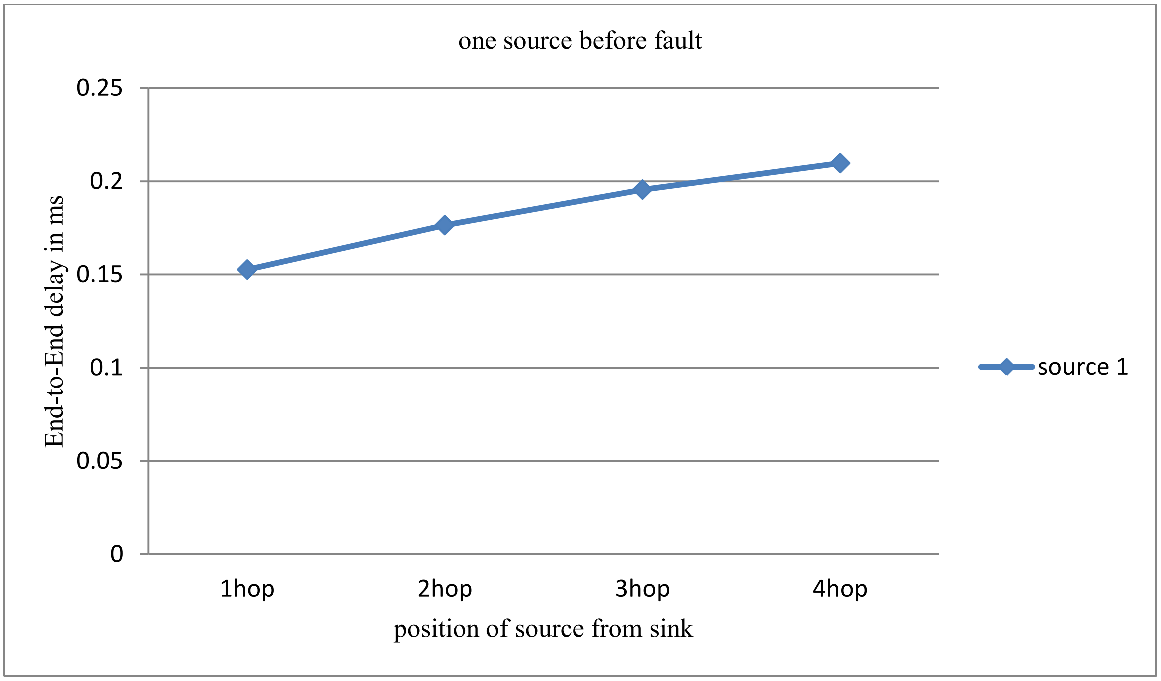
4.4.1. Scalability Test

The number of sensor nodes in WSNs is proportional with area (sensor field) needed to be controlled. Then when the sensor field is wider the follow-on increment of number of sensor nodes has a direct negative effect on reliability of the data transmission. In other words, when the number of sensor nodes in a given sensor field become many the traffic occurred among nodes will be higher than when the number of nodes in a sensor field are less. The follow-on collision among packets will let packets to be corrupted or lost. So the reliable protocol should withstand such type characteristics of WSNs and should recover the lost data if any. Particularly our protocol has been tested by increasing the number of sensor nodes up to 25 and by increasing the number of source nodes up to 3 that can possibly transmit at the same time. The following network performance figures show that the throughput and delay for end-to-end values. In the very beginning, we tested one source network before fault. When one source is transmitting a data to a sink, we saw that there is no data loss at all because there is no traffic and collision among packets. In order to see performance of our protocol, we used error model which drops a packet(s) that are being transmitted to the sink. So the simulation result before using error model, we called it before fault. And the result of one source network when using error model, we called it with fault.

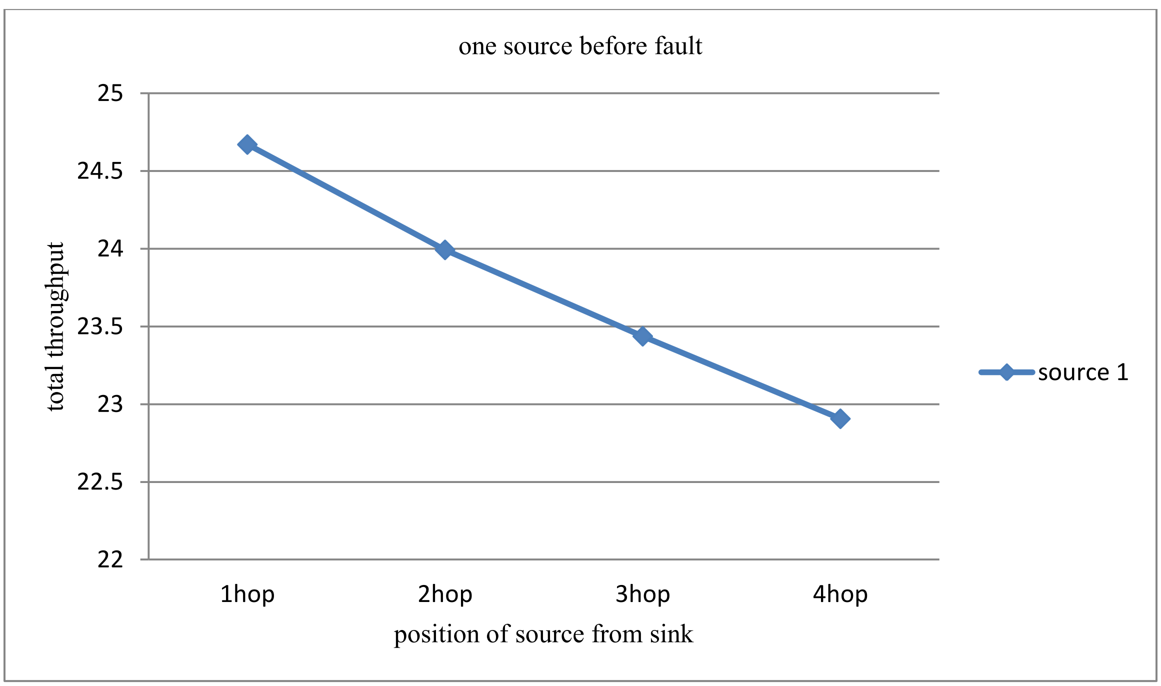
4.4.1.1. Single Source Single Sink

When a single source is sending to a single sink, no packet loss is observed in our simulation. This is because of there is no network congestion, no packet collision and as a result there is no packet or data loss. But in real world data/packet loss is not limited only if there is collision of packets but it is also may be data packets are corrupted. So test the performance of the new protocol whether the model can withstand such worst case problems, for this particular case we dropped packets deliberately that are being transmitted with some time intervals. Finally, the system has shown it is reliable and it can recover all the lost packets successfully.

The following first two figures show that the end-to-end delay and the total throughput of a single source network which is before fault. And the next two figures show that the aforementioned two performance matrices of a single source single sink network after packet drop is employed.



(a). End-to-end delay in msec.



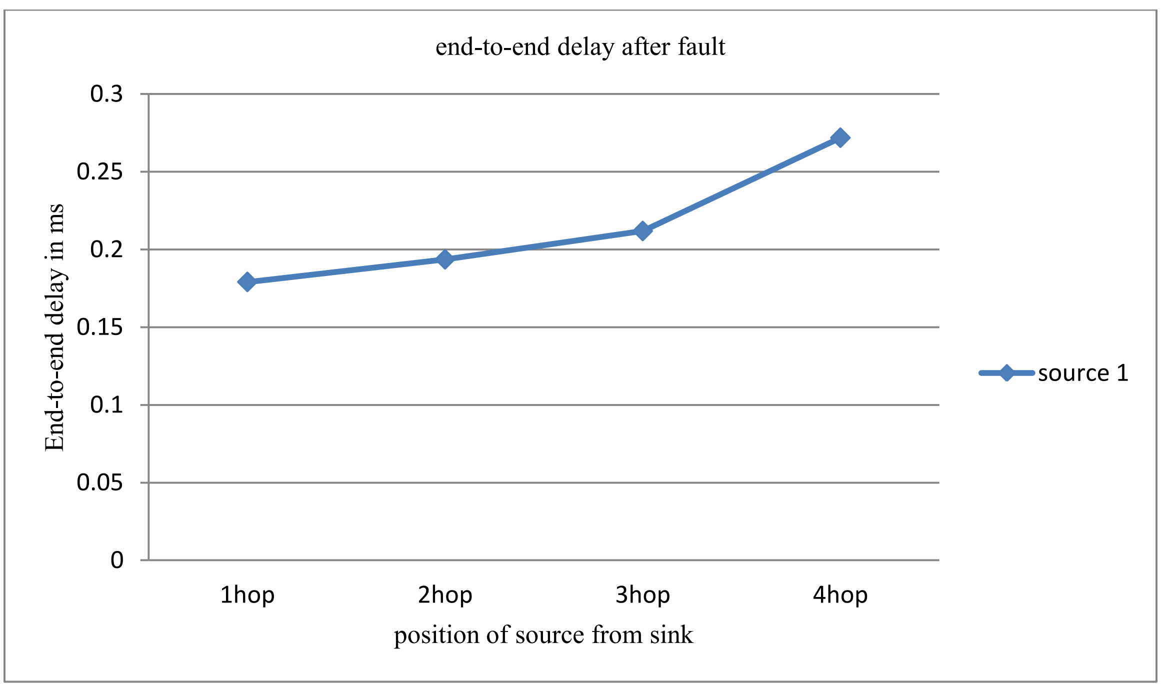
(b).total throughput

Figure4.3- Simulation result before using error model. (a) end-to-end delay in msec. (b) Total throughput

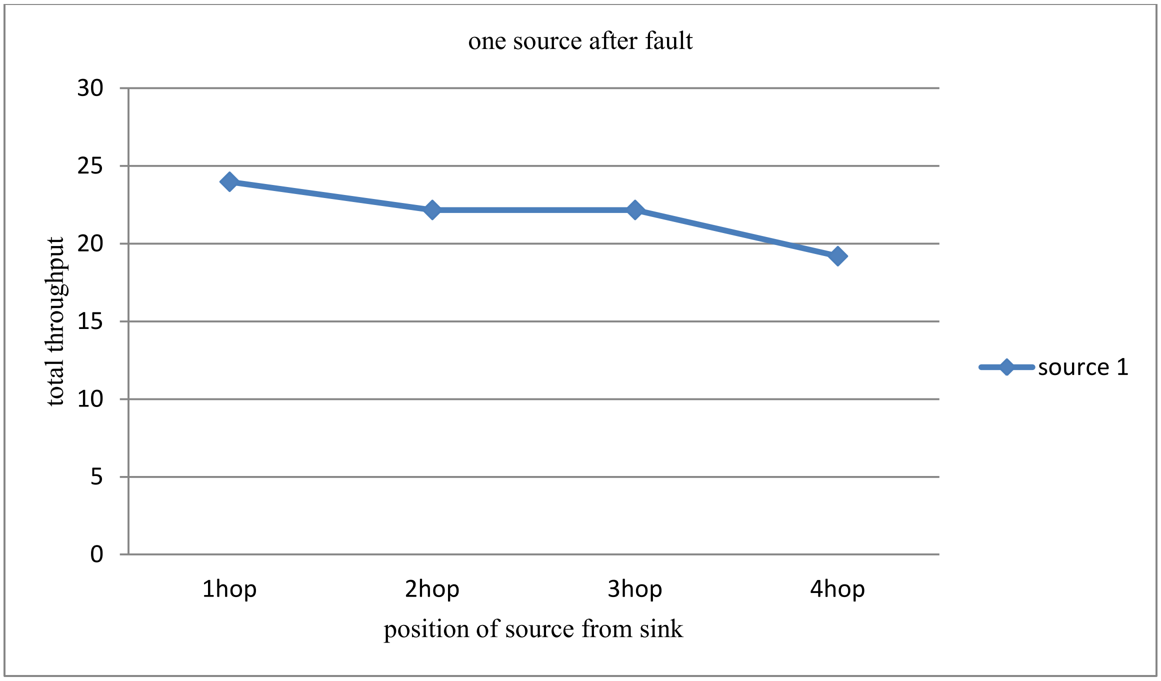
From above network performance figures we can observe that as the source node is further and further away from the sink the end-to-end delay increases linearly and the corresponding total throughput decreases linearly with negative slop.

Since end-to-end delay of a packet is the difference between its reception and sending times and this time difference is directly proportional with the distance that the packet covers, its value becomes bigger and bigger as the distance between the source and the sink is getting further and further apart. And as throughput is a total number of successfully delivered packets per the time difference between the reception time of the last packet and the sending time of the first packet, its value decreases as the this time difference value getting bigger.

On the other hand, when fault is injected to a network both the end-to-end delay and total throughput values depend on not only the distance between the source and the sink but also the time taken to recover a packet(s). That is all what we can observe from the following two figures.



(a) End-to-end delay in msec.



(b) Total throughput

Figure4.4- Simulation result after using error model (a) end-to-end delay in msec. (b) Total throughput

4.4.1.2. Multiple Sources Multiple Sinks

In the case of the presence of multiple sources send their data to their corresponding sinks there is no need to use error model because the atmosphere is exposed to packet collision. However, packet loss due to collision is not always happens when two or more sources send a data. So if needed to test the protocol when loss is always there, it is possible to use error model as an additional or expose the network for repeated collision by any other means. In this particular work, the protocol is tested when even neighboring nodes are sending and nodes sending in X-direction are sending. The following figures are for networks with two sources that can send their data at the same time. In here source 2 is when sink is node 19 and the possible sources are either node 15 or 14 or 13 or 12 for hop1, hop2, hop3 and hop4 respectively and source 1 is when sink is node 16 and the possible sources are either node 3 or 2 or 1 or 0 as can be seen from Figure4.5 NS2 NAM picture. Sensor nodes from 16 to 19 that are encircled by green lines are possible sinks for the simulation network.

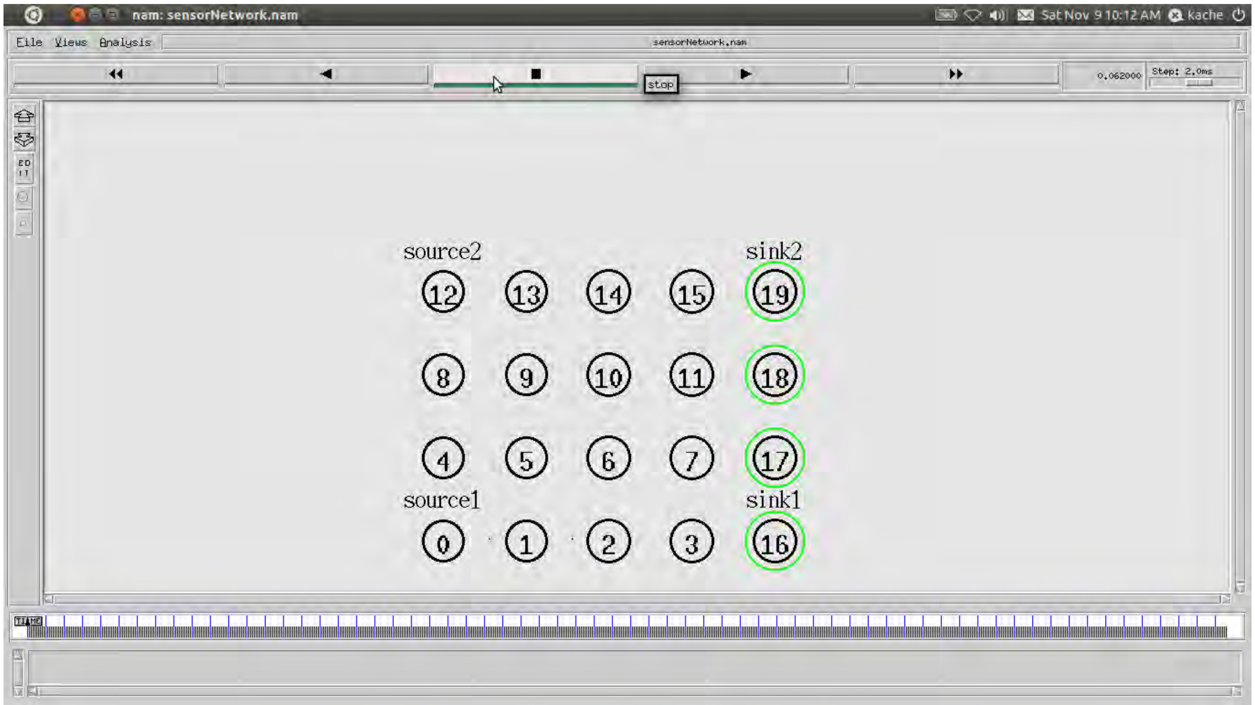
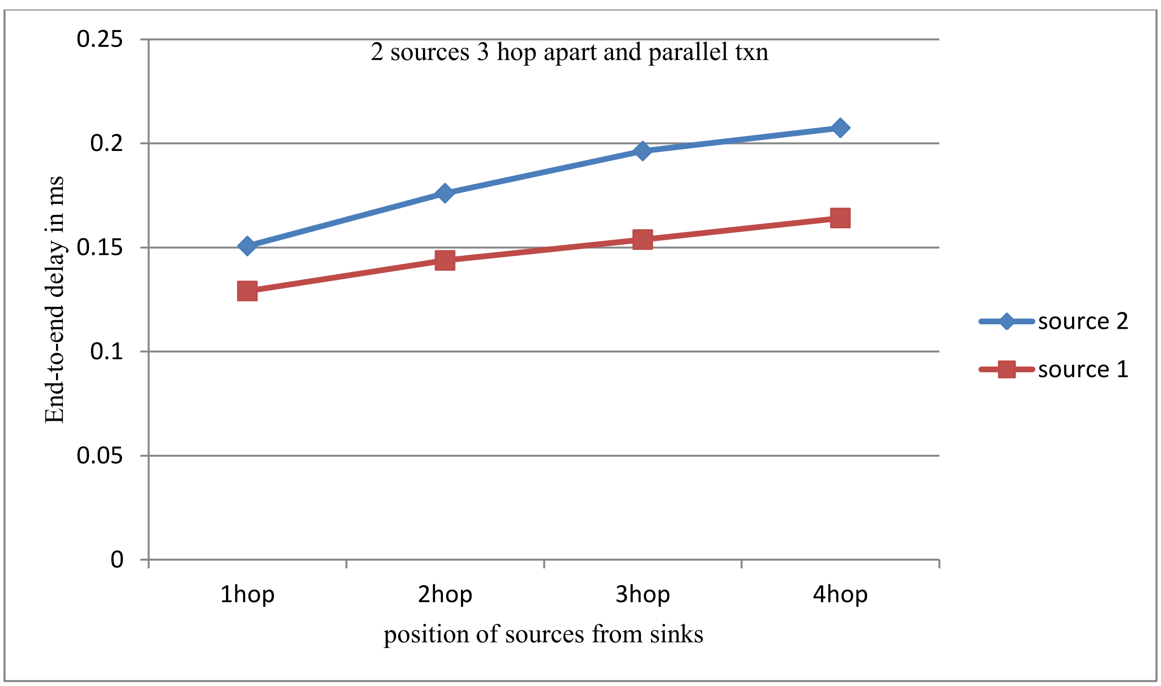
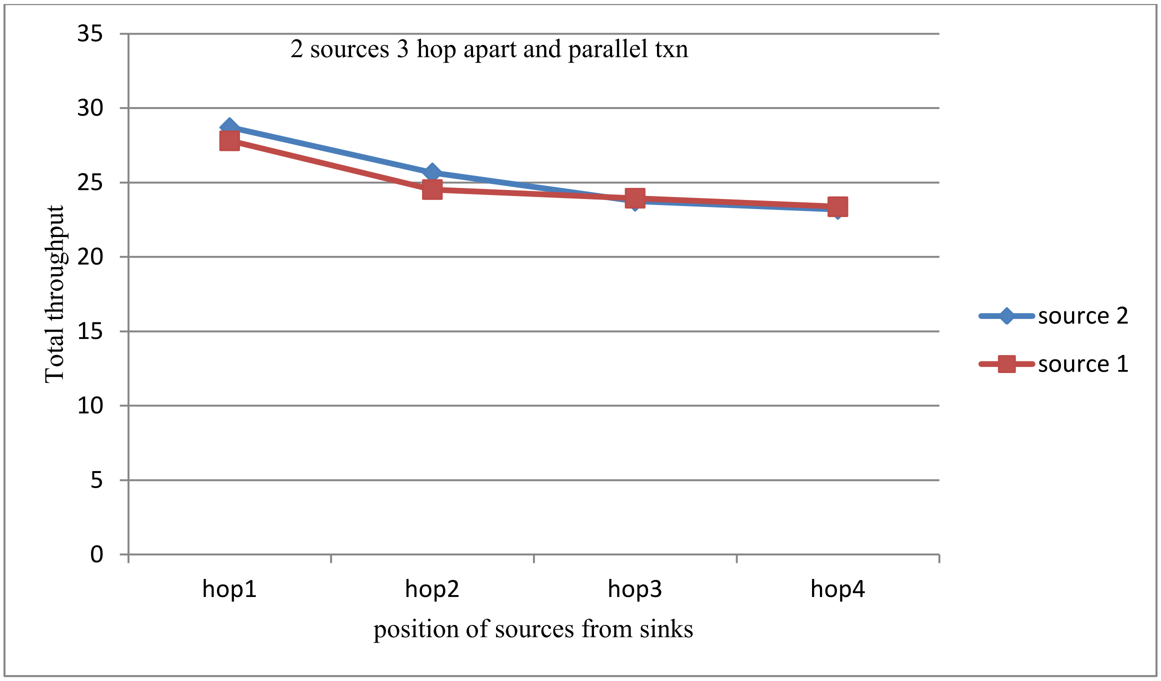


Figure4.5-Grid topology when there are two sources and two sinks that can transmit data at the same time from hop Four



(a) End-to-end delay in msec.

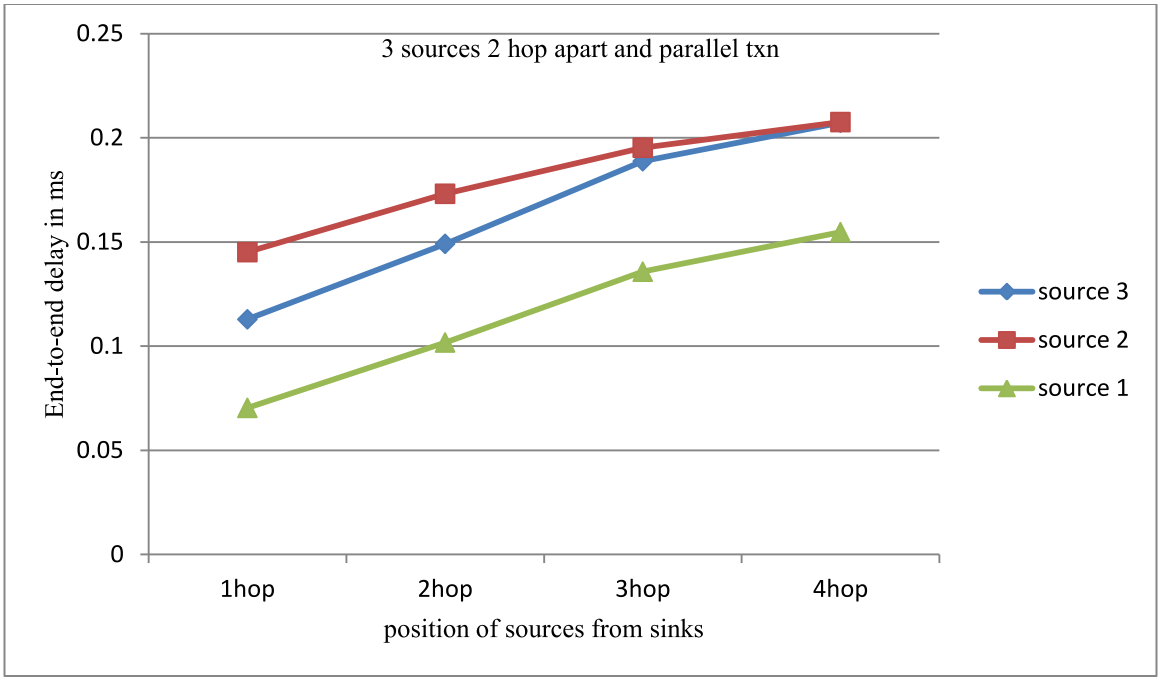


(b) Total throughput

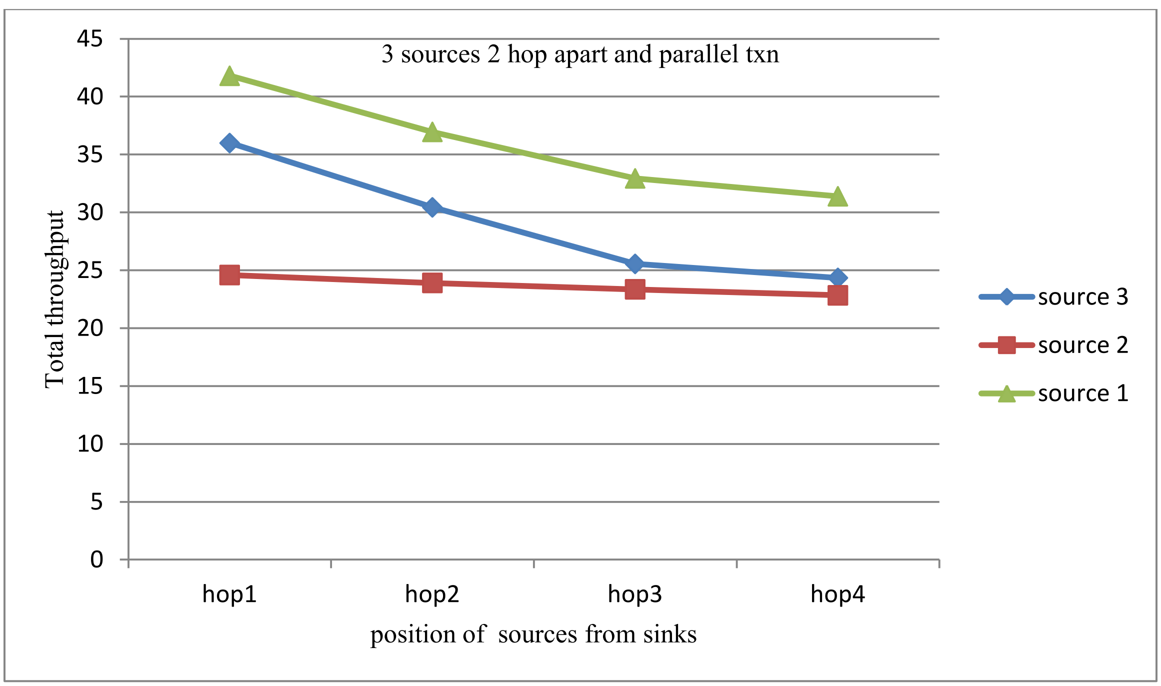
Figure4.6-Simulation result when there are two sources and two sinks. (a) end-to-end delay in msec. (b) Total throughput

By further increasing the number of sources to three in the network as can be seen in Figure4.2, we have tested the performance of our protocol i.e. its capability of error detection and recovery. And we analyzed that whatever the numbers of sources are in a network the reliability would be always 100% reliable. Figure4.7 show that the quality or performance of the network beside reliability.

In general, when two or more sensor source nodes are sending in parallel direction that are not neighbors there may or may not be a collision of data packets. This, on the other hand, means that packets may or may not lost or the frequency of occurrence is very small and it is observed during our simulation that its recovery is also very fast and the overall end-to-end delay is as small as required.



(a) End-to-end delay in ms



(b) Total throughput

Figure4. 7-Simulation result when there are three sources and three sinks. (a) end-to-end delay in msec. (b) Total throughput

When three sources are sending at the same time sometimes one of the three sources may wait until the other two completes transmitting their data successfully. This job is handled by a collision avoidance principle of MAC 802.11.

4.4.2. Interference and Collision Test

So far, we have done the effect of number of sources and the quality of our protocol. In here interference and collision tests are done by changing the direction of transmission and position of sending source nodes. When we were varying the number of source nodes, the gap among sources were two and three hops. For example, when the number of sources were two the vertical gap between sources were three hops and that of three sources were two hop apart vertically as shown in Figure4.5 and Figure4.2 respectively.

Figure4. 8-When source 1 (node 0) send to sink 1 (node 19) and source 2 (node 12) send sink 2 (node 16)

So when two neighboring nodes i.e. sources that are one hop apart, when two source nodes that are two and three hop apart vertically send at the same time are used to analyze the effect of position and number of source nodes in reliable data transmission. In the second part of this section we studied the effect of direction of transmission i.e. when two sources are sending to the X-direction as shown in Figure4.8. Source with address of 0 is sending to sink node with address of 19 and source node with address of 12 is sending to sink with address of 16.

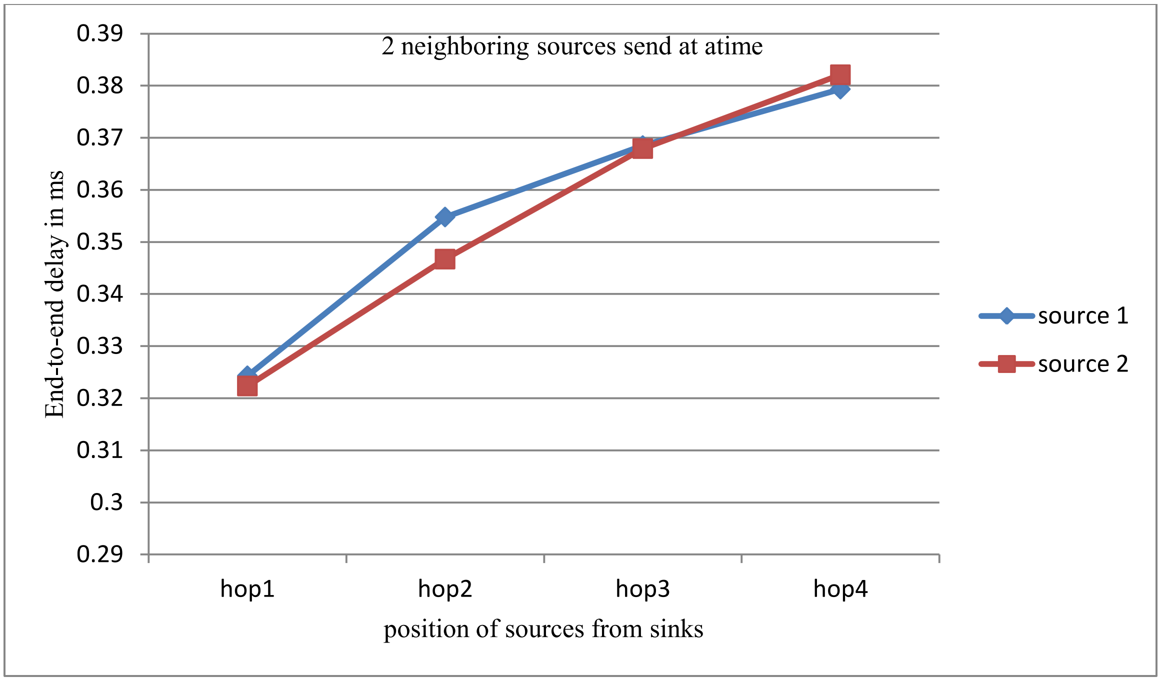
4.4.2.1. Varying Position of Sources

This and the following section are the worst cases that can challenge any protocol whether to be reliable or not. In here we let two neighboring nodes send at a time and during our simulation we observed four different cases for this particular section.

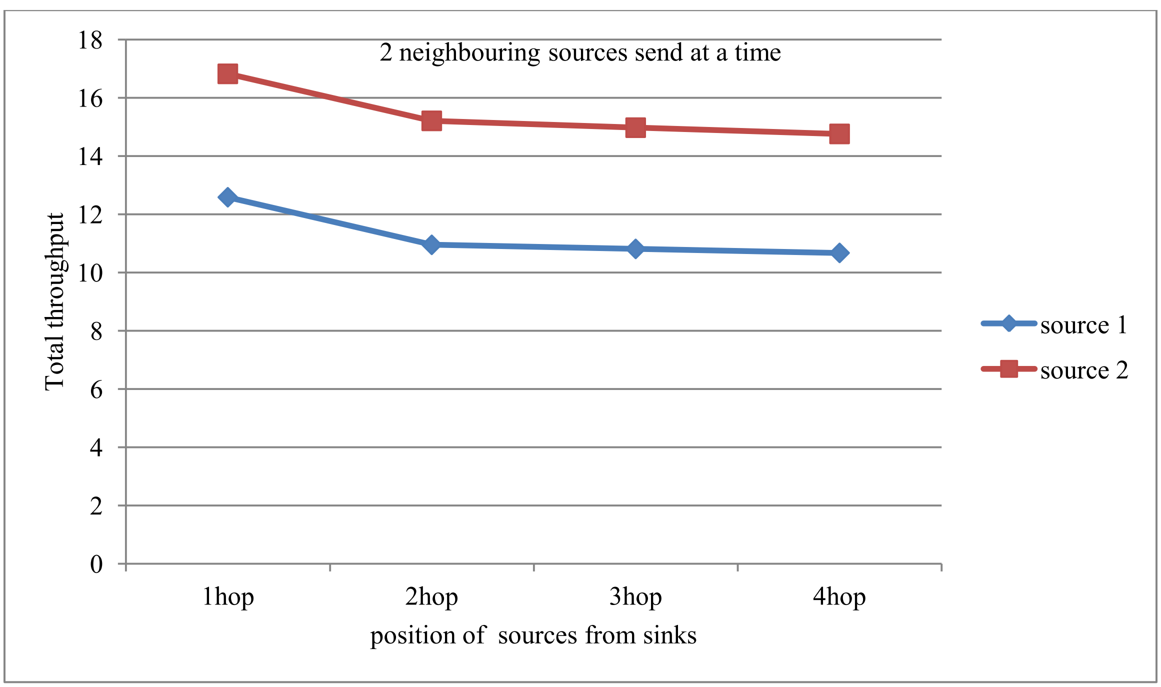
If there are other free sensor nodes that are other than the sending nodes and nodes in their path, the sending nodes mostly prefers to use that free nodes and try to reduce packet collision that would happen when they would be sending. In network types like in Figure4.8, for example, if node 0 and node 4 are sources to send their data to node 16 and 17 respectively, node 4 may use the path other than its row.

We restricted sending neighboring nodes not to use alternative path and we simulate the scenario with a two row network. After many repeated simulation results we could observe three different situations.

- Either one of the sending nodes waits and sends its data after another sending node successfully completes data transmission. In this condition most of the time the frequency of data loss is very small. Or,
- When one of the sending nodes is processing or doing something, the second node will start sending and when it is doing something other than transmitting packets may be recovering the lost data the former one will restart again.
- Or, in the worst case, when two sending nodes start sending at the same time, packet collision will be sever and the frequency of packet collision and recovery will be very high. However, the frequency of packet collision and recovery is very high it is still fully reliable.



(a) End-to-end delay in msec.



(b) Total throughput

Figure4. 9-Simulation result when there are two neighboring nodes are sending. (a) end-to-end delay in msec. (b) Total throughput

In general, the restricted type neighboring source nodes send at a time, the network performance is as shown below. And as expected from Figure4.9 (a) the end-to-end delay is very long time duration as compared with the end-to-end delays that have been seen in the previous sections. And from the same figure part (b) we can observe that the total throughput is very much smaller than the previous values of the same type.

4.4.2.2. Varying Direction of Transmission

As can be seen from Figure4.8 two sources are transmitting to the opposite direction (X-direction). In this type of transmission there are some nodes that receive packets from different nodes at the same time. Sensor nodes cannot receive packets from different nodes at the same time. If they try, or forcefully obliged to receive, there will be collision and as a result there will be a data loss. If this is the case, our protocol has to handle such type problems by detecting and retransmitting the lost packets. Data loss detection and recovery is done by checking each packet header that handles source address and their sequence number. In other words, the node at which packets of different sources pass through it, should check every packet's source node from their header and in order arrival of packets by their sequence number for the corresponding source. If there is a loss, it will automatically recover the lost data from the previous hop.

The reason why we don't draw some graphs that show the network performance during X-direction transmission is because, during simulation, it is observed that one of the sending nodes should wait until the other sending node completes transmission successfully. Because there will be one node in common for two paths if they attempt to transmit at the same time. If this is the case, almost all packets will be dropped. The protocol could detect but the there will be an all time job of loss and recovery. But thanks to the collision avoidance mechanism of MAC layer, only the first source which receives the CTS message will start sending and the second will be next after the sending node completes transmitting all data packets.

4.4.3. Life Time (Energy) Test

The experiment done in the previous sections was in a network that has all nodes active all the time. However, since battery of sensor nodes are not rechargeable and their life time is limited by the life time of their battery life, we used sleep scheduling technique to extend their life boundary. As described in the design section of sleep scheduling of chapter three, we used three sleep/wake cycles and the network is pre-scheduled in such a way that the area to be detected is fully covered and there is a full path (connected) from any active node to any sink. Full area coverage in a sensor field means there is no portion of area that is out of a sensing range of at least one sensor node at any time during scheduling. And full path transmission means, whenever a sensor node wants to transmit the sensed data to any sink, there is no need to wait the next node until it wakes up. It is already made to have a full path among all active nodes up to any sink.

Sink nodes are assumed as base stations and they are not limited in energy and other sources. They are active all the time to receive data from sensor nodes that are in all cycles.

Since the primary goal of this work is to design protocol for reliable data transmission of multi-source multi-sink WSNs by designing the most possible energy usage minimization technique. If any of the sensor nodes in a sensor field has data to be transmitted and if its' neighbors are asleep, there will be a data loss when any other node sends data to it while it is waiting to send its' own data to the sink because of buffer overflow. In addition to this when a sensor node waits to transmit a data until the next node wakes up, the end-to-end delay of data delivery will be very high. In other words, the action that would have been taken may be delayed and the intended objective will not be accomplished. For example, if the network is to detect fire in the forest, the action that may go to be taken may be delayed and forest may be fired. So 100% reliability without quick delivery of packets to sinks has no use if end-to-end delay is very high. To avoid such problem our protocol has full path from any source to any sink all the time.

Both of these protocols are developed in NS2 simulator and they work very well. The scheduling diagram is as shown below.

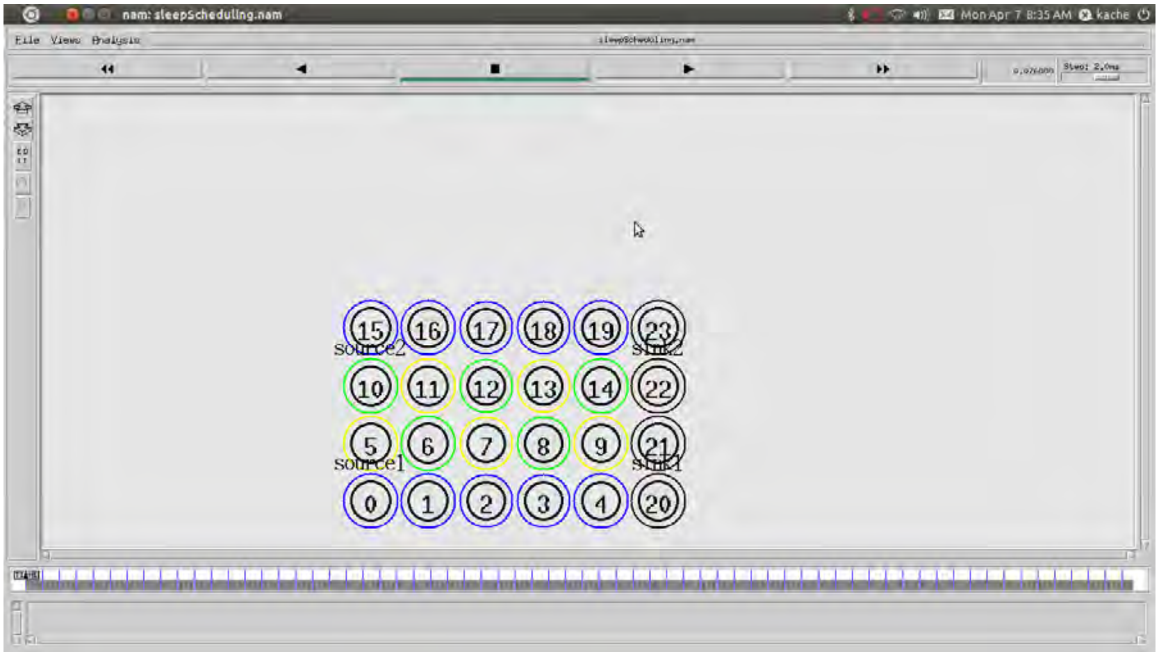


Figure4. 10-The sleep scheduling

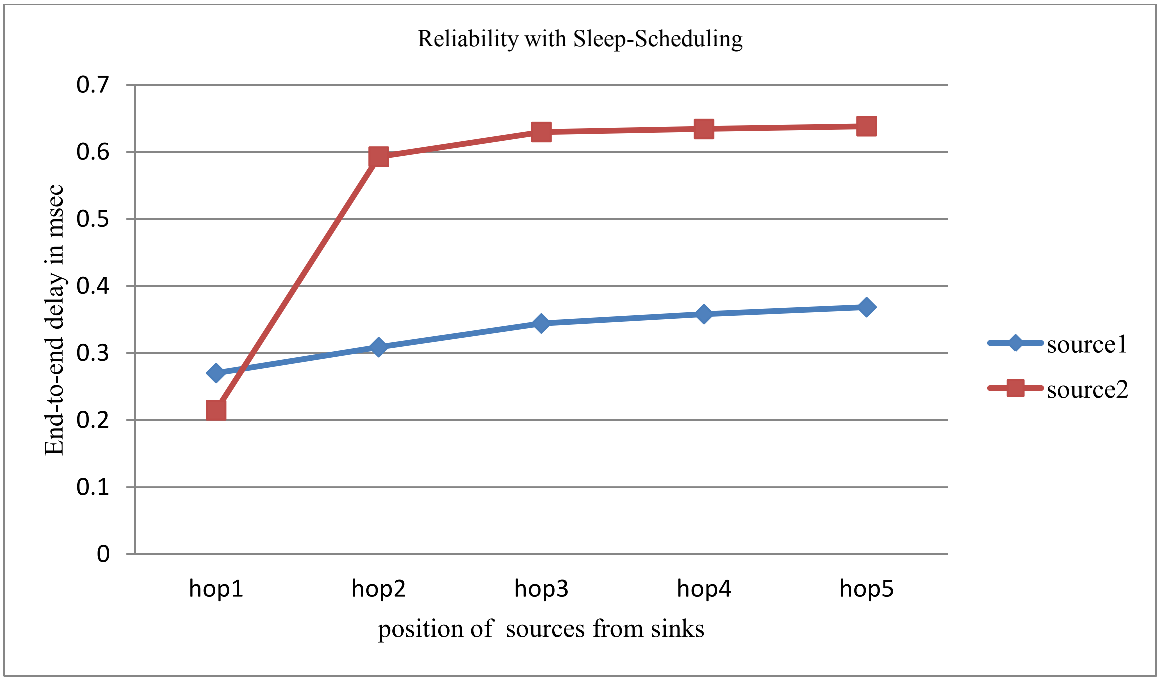
The reliable protocol that its performance already tested in the previous section and the sleep scheduling technique shown in Figure3.3 are developed to work together. Finally, the performance of the combined protocol is tested using different performance metrics such as increasing the number of source nodes that possibly send their data at the same time and by varying the position of sources. The worst case among these performance metrics is when neighboring sources or intermediates send at the same time. When two or more source nodes send in the X-direction, it is similar with what we have seen in the previous section i.e. one should wait until the other node that receives the CTS message before it receives completes transmission. So the performance of the above particular network is tested and shown below.

When source node 10 (source 2) and source node 0 (source 1) start sending to sink node 22 and node 20 respectively at the same time there are possible paths through which possible sending neighbors would occur. For example, if packets sourced from node 10 follows the path 10, 6, 12, 8, 14 and 22 and packets sourced from node 0 follows the path 0, 1, 2, 3, 4 and 20 (these number sequences are node IDs or node numbers), then node 6 with node 0, node 1 and node 2 and node 8 with node 2, node 3 and node 4 will be neighboring sending nodes. But during path construction before any of the sources start transmission: if there is a path that lets the two paths free from interference, it will be used by any one of the sending sources. For example, in the above network with the aforementioned scenario source 10 uses the path 10, 16, 17, 18, 19 and 22 and source 0 is as it was.

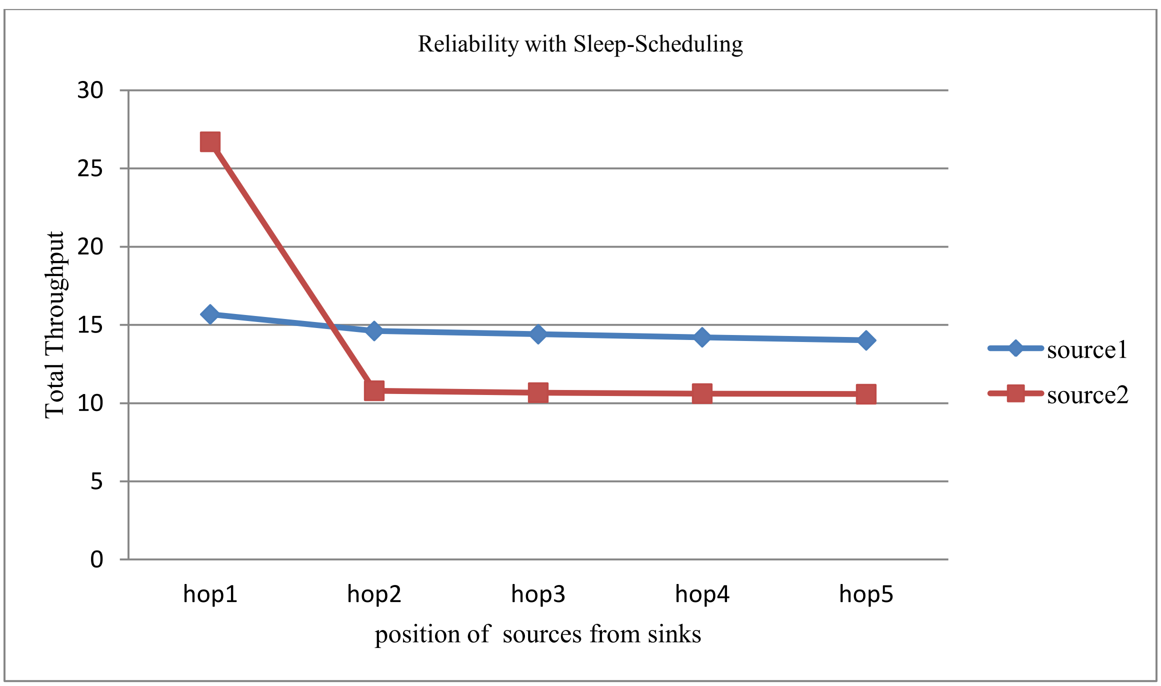
This is so when only source 10 starts sending after ½ T and its other corresponding time periods. Otherwise node 10 with node 0 and node 15 will never be active at the same time. The network performance shown below is by setting the value ½ T is equal to 3 seconds and letting sources (0 and 10) start sending at the same time i.e. exactly at 3.0 second. As already said the flow of packets of the second source follows the path which can avoid neighboring sending nodes.

The value of T (6.0 seconds) during simulation is given after observing many simulation completion times in order to identify the maximum time needed to complete packet transmission including when sources send tern by tern (the second source start sending after the first completes).

Since the network uses a shared medium, the behavior of transmission that is observed in here in some cases is similar with what have been seen in section 4.4.2.1 that is when the medium is sensed to be busy one node may wait until the other completes transmission or until it finds the medium free (possibly before the other sending node completes transmission). If the collision of data packets is occurred after the nodes start sending transmission, one of them will wait until its back off timer is zero. As the back off timer is equal to zero and the medium is free the node will start sending without caring whether the other node completes transmission. In such a way the nodes continue transmitting packets until they successfully completes transmission. This is the job of MAC protocol in particular MAC IEEE 802.11.



(a) End-to-end delay in msec.



(b) Total Throughput

Figure4. 11-Simulation results when reliable protocol is with sleep scheduling protocol. (a) end-to-end delay in msec. (b) Total throughput

Chapter Five

5. Conclusion and Future Work

5.1 Conclusion

WSNs have scarcity of many resources such as narrow bandwidth, small size and non-rechargeable battery, and small size memory are among the most usual characteristics of WSNs. As a result of that, there are issues that can be raised when we think about WSNs. Among them reliable data transmission and life time of WSNs are the very crucial and decisive issues. Many researchers work on these issues; however, when they were thinking about reliability they were not thinking about lifetime of sensor networks and vice versa. For example, in [1] reliable data transmission is done but they assumed that all sensor nodes are active all the time. In addition to that most of the time a type of network used for experimental analysis is single-source single-sink WSN. However, real world may be different.

Since WSNs are recent technologies [20] their deployment, energy usage, protocols used and others are not perfectly suited to them. Their energy supply, for example, is most of a time a non-rechargeable battery. Like any other wireless networks such as ad-hoc networks their bandwidth is also narrow; as a result reliable data transmission was difficult. Due to the above two most deficiencies of WSNs (non-rechargeable battery and reliability problems) increasing both energy efficiency and making data transmission reliable become a sensitive issue.

In order to increase the reliability of the network (in our protocol), hop-by-hop data loss detection and recovery is employed. Unlike end-to-end loss detection and recovery which is used in normal internet protocol (like TCP and some others), in every hop the receiving node will check whether the data is successfully receiving. If the receiver detects some data is lost, the recovery process will take place between two adjacent nodes i.e. the sending and the receiving node. Furthermore the NACK-based approach with the last single ACK packet is employed in order to fully detect the data loss. The former approach is the best to detect any data loss but its only drawback is when the whole data is lost. In this particular case the receiver even couldn't know whether packet(s) were sent to it. To avoid this problem the last single ACK packet is used. And that of energy efficiency is increased by reducing idle listening time of sensor nodes in a network of grid topology. Since most literatures tell us most of the energy of sensor nodes is consumed by idle listening i.e. listening packets that have not sent to them. So this unused waste of energy can be saved using pre-scheduled sleeping and waking up periods. All these ideas are implemented in NS2 simulator and some features are added to it (the simulator is extended) and the simulation and experimental results show that the new protocols perform well under various conditions and protocol parameter settings.

The simulation results show that the designed protocol is 100% reliable and the life time of a network is increased by around 100%. In addition to that by setting some protocol parameters the performance of the protocol under varies conditions is tested and it performs well. For example, the worst case for reliability problem of a grid topology network with multi-source multi-sink wireless sensor network is when neighboring sources are sending and when two sources are sending in X-direction. However, our protocol is even fully reliable in the aforementioned worst cases.

5.2 Future Work

This thesis work was only considering uni-cast transmission that is when multiple sources for multiple sinks or multiple sources to a single sink. But in real world, for a single source node, there could be multiple destinations of outgoing packets (multicasting). Therefore, in the future multicasting in multi-source multi-sink WSNs can be included.

The energy saving mechanism (pre-scheduling) of this paper is limited to a network of grid topology and in the future we recommend if energy efficient sleep scheduling is designed for a network of randomly scattered sensor nodes.

The value of T is given after observing repeated simulation. Since its time value should be sufficient to complete transmitting the sensed data at one time and it depends on the size of the network, it is recommended if there is a mechanism by which the network can determine the minimum value of T.

REFERENCES

[1]. Bofu yang, ”*Reliable Data Delivery in Wireless Sensor Networks*”, a thesis submitted to the college of graduate studies and research in partial fulfillment of the requirements for the degree of master of science in the department of Computer Science University of Saskatchewan, May 2010.

[2]. Christo Kleu,” An Ultra-Low Duty Cycle Sleep Scheduling Protocol Stack for Wireless Sensor Networks”, submitted in partial fulfillment of the requirement for the degree Master of Computer Engineering in University of Pretoria, March 2012.

[3]. Thijs Mutter, “Partition-based Network Load Balanced Routing in Large Scale Multi-sink Wireless Sensor Networks”, Department of Electrical Engineering, Mathematics and Computer Science, University of Twente, January 5, 2009.

[4]. Ian F. Akyildiz and Ismail H. Kasimoglu,” Wireless sensor and actor networks: research challenges”, School of Electrical and Computer Engineering, Georgia Institute of Technology, USA, May 2004.

[5]. Kay Romer,” Distributed Mining of Spatio-temporal Event Patterns in Sensor Networks”, Institute for Pervasive Computing ETH Zurich, Switzerland.

[6]. Sukun Kim, Rodrigo Fonseca and David Culler,” Reliable Transfer on Wireless Sensor Networks”, Electrical Engineering and Computer Sciences, University of California at Berkeley, California.

[7]. E. Felemban, C. Lee, E. Ekici, R. Boder, and S. Vural,” Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks”. In *Proc. IEEE INFOCOM ’05*, Miami, FL, pp. 2646--2657, Mar. 2005.

[8]. M. Khan, G. Pandurangan, and B. Bhargava, “Energy-Efficient Routing Schemes for Sensor Networks,” Technical Report CSD TR 03-013, Purdue University, July 2003.

[9]. Ademola P.Abidoye, Nureni A.Azeez, Ademola O.Adesina and Kehinde K.Agbele, “A Novel Routing Algorithm for Energy Optimization in Wireless Sensor Networks”, Research Journal of Information Technology 4(4):186-194, 2012.

[10]. David Shuman and Mingyan Liu, “Optimal Sleep Scheduling for a Wireless Sensor Network Node”, Electrical Engineering and Computer Science Department University of Michigan.

[11]. Kurtis Kredo II and Prasant Mohapatra, “Medium Access Control in Wireless Sensor Networks”, Electrical and Computer Engineering Department and Computer Science Department, University of California, June 2006.

[12]. Khayyat and A. Safwat, “Performance evaluation of distributed medium access schemes in multi-hop wireless sensor networks,” Tech. Rep., Queen’s University, Kingston, Canada, 2008, http://www.ece.queensu.ca/directory/faculty/Safwat.html.

[13]. F. Rocha. A. Grilo, P. R. Pereira, M. S. Nunes and A. Casaca, “Performance Evaluation of DTSN in Wireless Sensor Networks”, Lisboa, Portugal, pp. 1–9, 2008.

[14]. C.-Y. Wan, A. Campbell, and L. Krishnamurthy, “Pump-Slowly, Fetch-Quickly (PSFQ): A Reliable Transport Protocol for Sensor Networks”, WSNA 2002, Atlanta, Georgia, Sept. 2002.

[15]. M. H. Yaghmaee and D. Adjeroh, “A Reliable Transport Protocol for Wireless Sensor Networks”, West Virginia University, 2008 International Symposium on Telecommunications.

[16]. H. Zhang, A. Arora, Y.-R. Choi, and M. Gouda, “Reliable Bursty Convergecast in Wireless Sensor Networks”, Proc. ACM Mobihoc '05, Urbana-Champain, IL, May 25-28, 2005.

[17]. Wei Ye, John Heidemann, Deborah Estrin, “an energy efficient MAC protocol for wireless sensor network”, USC/ISI TECHNICAL REPORT ISI-TR-543.

[18]. Budhaditya Deb, Sudeept Bhatnagar and Badri Nath, “ReInForM: Reliable Information Forwarding Using Multiple Paths in Sensor Networks”, Dept. of Computer Science Rutgers University.

[19]. Fred Stann and John Heidemann, “RMST: Reliable Data Transport in Sensor Networks”, May 11, 2003.

[20]. Bobak Nazer and Michael Gastpar, “Reliable Physical Layer Network Coding”, arXiv: 1102.5724v1 [cs.IT], 28 Feb 2011.

[21]. Sukun Kim, Rodrigo Fonseca, Prabal Dutta, Arsalan Tavakoli, David Culler, Philip Levis, Scott Shenker, and Ion Stoica, *“Flush: A Reliable Bulk Transport Protocol*

for Multi-hop Wireless Networks", Computer Science Division in University of California, and Computer Systems Lab in Stanford University.

[22]. Yao-Nan Lien, "Hop-by-Hop TCP for Sensor Networks", IJCNC, Vol.1, No.1, April 2009.

[23]. Gang Lu, Narayanan Sadagopan, Bhaskar Krishnamachari and Ashish Goel,*" Delay Efficient Sleep Scheduling in Wireless Sensor Networks"*, Department of Electrical Engineering, Department of Computer Science, University of Southern California and Department of Management Science and Engineering, Stanford University.

[24]. Teerawat Issariyakul and Ekram Hossain, "Introduction to Network Simulator NS2 1^{st}&2^{nd} ed.", DOI 10.1007/978-1-4614-1406-3, Springer Science+Business Media, LLC 2012.

[25]. Junchao Ma, Wei Lou, Yanwei Wu, Xiang-Yang Li and Guihai Chen "Energy Efficient TDMA Sleep Scheduling in Wireless Sensor Networks", in Proc. of IEEE INFOCOM, 2009.

[26]. Pietro Ciciriello, Luca Mottola, and Gian Pietro Picco," Efficient Routing from Multiple Sources to Multiple Sinks in Wireless Sensor Networks", department of electronics and information and department of information and communication technology, Italy.

[27]. Ian T. Downard, "Simulating Sensor Networks in NS-2", Naval Research Laboratory, May 31, 2004.

APPENDIX-A

SAMPLE SOURCE CODES

```
#=================================================================
# CAUTION! Duplicating or using this source code without the permission of the
# programmer is illegal.
# This programme simulates wireless sensor networks when there are multiple sources
# multiple sinks. At a time two randomly selected sources are designed to transmit
# to two sinks.
#=================================================================
set val(chan) Channel/WirelessChannel ;# channel type
set val(prop) Propagation/TwoRayGround ;# radio-propagation model
set val(netif) Phy/WirelessPhy ;# network interface type
set val(mac) Mac/802_11 ;# MAC type
set val(ifq) Queue/DropTail/PriQueue ;# interface queue type
set val(ll) LL ;# link layer type
set val(ant) Antenna/OmniAntenna ;# antenna model
set val(ifqlen) 500 ;# max packet in ifq
set val(nn) 20 ;# number of mobilenodes
set val(rp) AODV ;# routing protocol
set val(x) 1000 ;# X dimension of topography
set val(y) 1000 ;# Y dimension of topography
set val(stop) 150 ;# time of simulation end
#=================================================================
# creating simulator object
#=================================================================
set ns [new Simulator]
set tracefd [open sensorNetwork.tr w]
set namtrace [open sensorNetwork.nam w]
```

```
#======================================
# creating trace file
#======================================
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
#======================================
# set up topography object
#======================================
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
#======================================
# Creating God
#======================================
create-god $val(nn)
#==================================================================
# configure the nodes
#==================================================================
$ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace OFF \
```

```
for {set i 0} {$i < $val(nn) } { incr i } {
set n($i) [$ns node]
}
#===================================================================
# Provide initial location of mobilenodes
#===================================================================
$n(0) set X_ 0.0
$n(0) set Y_ 0.0
$n(0) set Z_ 0.0

$n(1) set X_ 200.0
$n(1) set Y_ 0.0
$n(1) set Z_ 0.0

$n(2) set X_ 400.0
$n(2) set Y_ 0.0
$n(2) set Z_ 0.0

$n(3) set X_ 600.0
$n(3) set Y_ 0.0
$n(3) set Z_ 0.0

$n(4) set X_ 0.0
$n(4) set Y_ 200.0
$n(4) set Z_ 0.0

$n(5) set X_ 200.0
$n(5) set Y_ 200.0
$n(5) set Z_ 0.0
```

```
$n(6) set X_ 400.0
$n(6) set Y_ 200.0
$n(6) set Z_ 0.0

$n(7) set X_ 600.0
$n(7) set Y_ 200.0
$n(7) set Z_ 0.0

$n(8) set X_ 0.0
$n(8) set Y_ 400.0
$n(8) set Z_ 0.0

$n(9) set X_ 200.0
$n(9) set Y_ 400.0
$n(9) set Z_ 0.0

$n(10) set X_ 400.0
$n(10) set Y_ 400.0
$n(10) set Z_ 0.0

$n(11) set X_ 600.0
$n(11) set Y_ 400.0
$n(11) set Z_ 0.0

$n(12) set X_ 0.0
$n(12) set Y_ 600.0
$n(12) set Z_ 0.0

$n(13) set X_ 200.0
$n(13) set Y_ 600.0
$n(13) set Z_ 0.0
```

```
$n(14) set X_ 400.0
$n(14) set Y_ 600.0
$n(14) set Z_ 0.0

$n(15) set X_ 600.0
$n(15) set Y_ 600.0
$n(15) set Z_ 0.0

$n(16) set X_ 800.0
$n(16) set Y_ 0.0
$n(16) set Z_ 0.0

$n(17) set X_ 800.0
$n(17) set Y_ 200.0
$n(17) set Z_ 0.0

$n(18) set X_ 800.0
$n(18) set Y_ 400.0
$n(18) set Z_ 0.0

$n(19) set X_ 800.0
$n(19) set Y_ 600.0
$n(19) set Z_ 0.0

#===================================================================
# Random number generation so that we could have random sources for every
#simulation time
#===================================================================
$defaultRNG seed 0
set x 0
```

```
set y 0
set source1RNG [new RNG]
set source2RNG [new RNG]
set destRNG [new RNG]

set source1_ [new RandomVariable/Uniform]
$source1_ set min_ 1
$source1_ set max_ 5000
$source1_ use-rng $source1RNG

set source2_ [new RandomVariable/Uniform]
$source2_ set min_ 1
$source2_ set max_ 5000
$source2_ use-rng $source2RNG

#puts "random numbers"
#set x [expr round([$source1_ value])%16]
#set y [expr round([$source2_ value])%16]

puts "random source numbers are"
set x [expr round([$source1_ value])%16]
set y [expr round([$source2_ value])%16]

if {$x==$y} {
  set x [expr $x+1]
  if {$x==16} {
      set x [expr $x-2]
    }
}
puts $x
puts $y
```

```
#==============================================================
# Set a UDP connection between arbitrary source and n(17)
#==============================================================
set udp1 [new Agent/UDP]
#$ns attach-agent $n($x) $udp1
$ns attach-agent $n(12) $udp1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 1000
$cbr1 set maxpkts_ 10
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

set null [new Agent/Null]
$ns attach-agent $n(19) $null

$ns connect $udp1 $null

$ns at 0.1 "$cbr1 start"
#$ns at 5.0 "$cbr1 send 100000"
#==============================================================
# Set a UDP connection between arbitrary source and n(16) but this source node # is deferent from
# the above source even though they are selected arbitrarily they are made not to coincide.
#==============================================================
set udp2 [new Agent/UDP]
#$ns attach-agent $n($y) $udp
$ns attach-agent $n(0) $udp2

set cbr2 [new Application/Traffic/CBR]
```

```
$cbr2 set interval_ 0.005
$cbr2 set maxpkts_ 10
$cbr2 set packetSize_ 1000
$cbr2 attach-agent $udp2

set null [new Agent/Null]
$ns attach-agent $n(16) $null

$ns connect $udp2 $null

$ns at 0.0 "$cbr2 start"
#$ns at 5.0 "$cbr2 send 100000"
#================================================================
#defining heads
#================================================================
#$ns at 0.0 "$n($x) label source1"
$ns at 0.0 "$n(0) label source1"
$ns at 0.0 "$n(16) label  sink1"
#$ns at 0.0 "$n($y) label source2"
$ns at 0.0 "$n(12) label source2"
$ns at 0.0 "$n(19)  label sink2"
#================================================================
#color given to destinations
#================================================================
$ns at 0.0 "$n(16) add-mark N16 Green circle"
$ns at 0.0 "$n(17) add-mark N17 Green circle"
$ns at 0.0 "$n(18) add-mark N18 Green circle"
$ns at 0.0 "$n(19) add-mark N19 Green circle"
#================================================================
# Define node initial position in nam
#================================================================
```

```
for {set i 0} {$i < $val(nn)} { incr i } {
# 100 defines the node size for nam
$ns initial_node_pos $n($i) 100
}
#==================================================================
# Telling nodes when the simulation ends
#==================================================================
for {set i 0} {$i < $val(nn) } { incr i } {
$ns at $val(stop) "$n($i) reset";
}
#==================================================================
# ending nam and the simulation
#==================================================================
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 150.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
global ns tracefd namtrace
$ns flush-trace
close $tracefd
close $namtrace
exec nam sensorNetwork.nam &
}
#==================================================================
# Make the script to execute
#==================================================================
$ns run
```

APPENDIX-B

SAMPLE RESULTS

random source numbers are

13

8

channel.cc:sendUp - Calc highestAntennaZ_ and distCST_

highestAntennaZ_ = 1.5, distCST_ = 550.0

SORTING LISTS ...DONE!

uid 0 sender: 1 at: 0.030813

time: 0.040574 first: 0 seqno: 2 psize: 0 name: cbr direction: 1 npkt: 2 uid: 0 saddr: 1 this node is: 2

uid 0 sender: 2 at: 0.040599

uid 1 sender: 1 at: 0.040813

uid 2 sender: 1 at: 0.047619

time: 0.050460 first: 0 seqno: 3 psize: 0 name: cbr direction: 1 npkt: 2 uid: 0 saddr: 1 this node is: 3

uid 0 sender: 3 at: 0.050485

time: 0.060386 first: 0 seqno: 3 psize: 0 name: cbr direction: 1 npkt: 2 uid: 0 saddr: 1 this node is: 16

time: 0.070163 first: 0 seqno: 3 psize: 0 name: cbr direction: 1 npkt: 2 uid: 1 saddr: 1 this node is: 2

uid 1 sender: 2 at: 0.070188

uid 3 sender: 1 at: 0.071429

time: 0.080269 first: 0 seqno: 4 psize: 0 name: cbr direction: 1 npkt: 3 uid: 2 saddr: 1 this node is: 2

uid 2 sender: 2 at: 0.080294

time: 0.090055 first: 0 seqno: 4 psize: 0 name: cbr direction: 1 npkt: 3 uid: 1 saddr: 1 this node is: 3

uid 1 sender: 3 at: 0.090080

uid 4 sender: 1 at: 0.095238

time: 0.100041 first: 0 seqno: 4 psize: 0 name: cbr direction: 1 npkt: 3 uid: 1 saddr: 1 this node is: 16

time: 0.109958 first: 0 seqno: 5 psize: 0 name: cbr direction: 1 npkt: 4 uid: 3 saddr: 1 this node is: 2

uid 3 sender: 2 at: 0.109983

uid 9 sender: 1 at: 0.119048

time: 0.119664 first: 0 seqno: 5 psize: 0 name: cbr direction: 1 npkt: 4 uid: 2 saddr: 1 this node is: 3

uid 2 sender: 3 at: 0.119689

uid 5 sender: 13 at: 0.126499

time: 0.129330 first: 0 seqno: 5 psize: 0 name: cbr direction: 1 npkt: 4 uid: 2 saddr: 1 this node is: 16

uid 12 sender: 13 at: 0.130000

uid 13 sender: 13 at: 0.135000

uid 6 sender: 13 at: 0.136499

time: 0.137389 first: 0 seqno: 3 psize: 0 name: cbr direction: 1 npkt: 2 uid: 5 saddr: 13 this node is: 14

uid 5 sender: 14 at: 0.137414

time: 0.139421 first: 0 seqno: 6 psize: 0 name: cbr direction: 1 npkt: 5 uid: 3 saddr: 1 this node is: 3

uid 3 sender: 3 at: 0.139446

uid 14 sender: 13 at: 0.140000

uid 15 sender: 1 at: 0.142857

uid 16 sender: 13 at: 0.145000

uid 7 sender: 13 at: 0.146499

time: 0.147195 first: 0 seqno: 4 psize: 0 name: cbr direction: 1 npkt: 4 uid: 5 saddr: 13 this node is: 15

uid 5 sender: 15 at: 0.147220

time: 0.149187 first: 0 seqno: 6 psize: 0 name: cbr direction: 1 npkt: 5 uid: 3 saddr: 1 this node is: 16

uid 8 sender: 13 at: 0.156499

time: 0.157041 first: 0 seqno: 4 psize: 0 name: cbr direction: 1 npkt: 4 uid: 5 saddr: 13 this node is: 19

time: 0.159867 first: 0 seqno: 6 psize: 0 name: cbr direction: 1 npkt: 5 uid: 4 saddr: 1 this node is: 2

uid 4 sender: 2 at: 0.159892

uid 10 sender: 13 at: 0.166499

uid 17 sender: 1 at: 0.166667

time: 0.166937 first: 0 seqno: 4 psize: 0 name: cbr direction: 1 npkt: 3 uid: 12 saddr: 13 this node is: 14

uid 12 sender: 14 at: 0.166962

time: 0.169633 first: 0 seqno: 7 psize: 0 name: cbr direction: 1 npkt: 6 uid: 9 saddr: 1 this node is: 2

uid 9 sender: 2 at: 0.169658

uid 11 sender: 13 at: 0.176499

time: 0.176784 first: 0 seqno: 5 psize: 0 name: cbr direction: 1 npkt: 4 uid: 13 saddr: 13 this node is: 14

uid 13 sender: 14 at: 0.176809

time: 0.179499 first: 0 seqno: 7 psize: 0 name: cbr direction: 1 npkt: 6 uid: 4 saddr: 1 this node is: 3

uid 4 sender: 3 at: 0.179524

time: 0.186590 first: 0 seqno: 5 psize: 0 name: cbr direction: 1 npkt: 4 uid: 12 saddr: 13 this node is: 15

uid 12 sender: 15 at: 0.186615

time: 0.189265 first: 0 seqno: 7 psize: 0 name: cbr direction: 1 npkt: 6 uid: 4 saddr: 1 this node is: 16

uid 18 sender: 1 at: 0.190476

time: 0.196336 first: 0 seqno: 5 psize: 0 name: cbr direction: 1 npkt: 4 uid: 12 saddr: 13 this node is: 19

time: 0.200181 first: 0 seqno: 8 psize: 0 name: cbr direction: 1 npkt: 7 uid: 15 saddr: 1 this node is: 2

uid 15 sender: 2 at: 0.200206

time: 0.206293 first: 0 seqno: 6 psize: 0 name: cbr direction: 1 npkt: 5 uid: 6 saddr: 13 this node is: 14

uid 6 sender: 14 at: 0.206318

time: 0.210866 first: 0 seqno: 8 psize: 0 name: cbr direction: 1 npkt: 7 uid: 9 saddr: 1 this node is: 3

uid 9 sender: 3 at: 0.210891

uid 19 sender: 1 at: 0.214286

time: 0.216299 first: 0 seqno: 6 psize: 0 name: cbr direction: 1 npkt: 5 uid: 13 saddr: 13 this node is: 15

uid 13 sender: 15 at: 0.216324

time: 0.220632 first: 0 seqno: 8 psize: 0 name: cbr direction: 1 npkt: 7 uid: 9 saddr: 1 this node is: 16

time: 0.225985 first: 0 seqno: 6 psize: 0 name: cbr direction: 1 npkt: 5 uid: 13 saddr: 13 this node is: 19

time: 0.230749 first: 0 seqno: 9 psize: 0 name: cbr direction: 1 npkt: 8 uid: 17 saddr: 1 this node is: 2

uid 17 sender: 2 at: 0.230774

time: 0.235881 first: 0 seqno: 7 psize: 0 name: cbr direction: 1 npkt: 6 uid: 14 saddr: 13 this node is: 14

uid 14 sender: 14 at: 0.235906

node:1! please resend packet number:10 i am node:2

time: 0.245827 first: 0 seqno: 7 psize: 0 name: cbr direction: 1 npkt: 6 uid: 6 saddr: 13 this node is: 15

uid 6 sender: 15 at: 0.245852

node:1! please resend packet number:10 i am node:2

time: 0.255633 first: 0 seqno: 7 psize: 0 name: cbr direction: 1 npkt: 6 uid: 6 saddr: 13 this node is: 19

time: 0.260846 first: 0 seqno: 9 psize: 0 name: cbr direction: 1 npkt: 8 uid: 15 saddr: 1 this node is: 3

uid 15 sender: 3 at: 0.260871

time: 0.265550 first: 0 seqno: 8 psize: 0 name: cbr direction: 1 npkt: 7 uid: 16 saddr: 13 this node is: 14

uid 16 sender: 14 at: 0.265575

node:2! please resend packet number:10 i am node:3

time: 0.275656 first: 0 seqno: 8 psize: 0 name: cbr direction: 1 npkt: 7 uid: 14 saddr: 13 this node is: 15

uid 14 sender: 15 at: 0.275681

node:3! please resend packet number:9 i am node:16

time: 0.285363 first: 0 seqno: 9 psize: 0 name: cbr direction: 1 npkt: 8 uid: 16 saddr: 13 this node is: 15

uid 16 sender: 15 at: 0.285388

sorry nack is received in:1

nack seqno:9

ok i am sending a packet with seq= 10 for the 2nd time ...

sorry nack is received in:1

nack seqno:9

ok i am sending a packet with seq= 10 for the 2nd time ...

sorry nack is received in:2

nack seqno:9

ok i am sending a packet with seq= 10 for the 2nd time ...

time: 0.292520 first: 0 seqno: 10 psize: 0 name: cbr direction: 1 npkt: 9 uid: 0 saddr: 1 this node is: 2

uid 0 sender: 2 at: 0.292545

time: 0.294046 first: 0 seqno: 10 psize: 0 name: cbr direction: 1 npkt: 9 uid: 0 saddr: 1 this node is: 3

uid 0 sender: 3 at: 0.294071

time: 0.295449 first: 0 seqno: 8 psize: 0 name: cbr direction: 1 npkt: 7 uid: 14 saddr: 13 this node is: 19

sorry nack is received in:3

nack seqno:8

ok i am sending a packet with seq= 9 for the 2nd time ...

node:2! please resend packet number:11 i am node:3

node:3! please resend packet number:9 i am node:16

time: 0.301698 first: 0 seqno: 9 psize: 0 name: cbr direction: 1 npkt: 8 uid: 0 saddr: 1 this node is: 16

sorry nack is received in:2

nack seqno:10

ok i am sending a packet with seq= 11 for the 2nd time ...

time: 0.305235 first: 0 seqno: 9 psize: 0 name: cbr direction: 1 npkt: 8 uid: 16 saddr: 13 this node is: 19

time: 0.305391 first: 0 seqno: 11 psize: 0 name: cbr direction: 1 npkt: 10 uid: 0 saddr: 1 this node is: 3

uid 0 sender: 3 at: 0.305416

sorry nack is received in:3

nack seqno:8

ok i am sending a packet with seq= 9 for the 2nd time ...

======================================

congradulation! ack is received in:2

======================================

node:3! please resend packet number:10 i am node:16

sorry nack is received in:3

nack seqno:9

ok i am sending a packet with seq= 10 for the 2nd time ...

node:3! please resend packet number:10 i am node:16

time: 0.315032 first: 0 seqno: 9 psize: 0 name: cbr direction: 1 npkt: 8 uid: 7 saddr: 13 this node is: 14

uid 7 sender: 14 at: 0.315057

sorry nack is received in:3

nack seqno:9

ok i am sending a packet with seq= 10 for the 2nd time ...

time: 0.318226 first: 0 seqno: 10 psize: 0 name: cbr direction: 1 npkt: 9 uid: 0 saddr: 1 this node is: 16

node:3! please resend packet number:11 i am node:16

sorry nack is received in:3

nack seqno:10

ok i am sending a packet with seq= 11 for the 2nd time ...

time: 0.324094 first: 0 seqno: 11 psize: 0 name: cbr direction: 1 npkt: 10 uid: 0 saddr: 1 this node is: 16

time: 0.324739 first: 0 seqno: 10 psize: 0 name: cbr direction: 1 npkt: 9 uid: 8 saddr: 13 this node is: 14

uid 8 sender: 14 at: 0.324764

=======================================

congradulation! ack is received in:3

=======================================

node:1! please resend packet number:11 i am node:2

sorry nack is received in:1

nack seqno:10

ok i am sending a packet with seq= 11 for the 2nd time ...

time: 0.332883 first: 0 seqno: 11 psize: 0 name: cbr direction: 1 npkt: 10 uid: 0 saddr: 1 this node is: 2

uid 0 sender: 2 at: 0.332908

time: 0.334505 first: 0 seqno: 11 psize: 0 name: cbr direction: 1 npkt: 10 uid: 10 saddr: 13 this node is: 14

uid 10 sender: 14 at: 0.334530

=======================================

congradulation! ack is received in:1

=======================================

node:2! please resend packet number:12 i am node:3

sorry nack is received in:2

nack seqno:11

ok i am sending a packet with seq= 12 for the 2nd time ...

time: 0.339848 first: 0 seqno: 12 psize: 0 name: cbr direction: 1 npkt: 11 uid: 0 saddr: 1 this node is: 3

uid 0 sender: 3 at: 0.339873

node:3! please resend packet number:12 i am node:16

sorry nack is received in:3

nack seqno:11

ok i am sending a packet with seq= 12 for the 2nd time ...

time: 0.344191 first: 0 seqno: 10 psize: 0 name: cbr direction: 1 npkt: 9 uid: 7 saddr: 13 this node is: 15

uid 7 sender: 15 at: 0.344216

time: 0.345506 first: 0 seqno: 12 psize: 0 name: cbr direction: 1 npkt: 11 uid: 0 saddr: 1 this node is: 16

time: 0.354297 first: 0 seqno: 10 psize: 0 name: cbr direction: 1 npkt: 9 uid: 7 saddr: 13 this node is: 19

time: 0.364234 first: 0 seqno: 12 psize: 0 name: cbr direction: 1 npkt: 11 uid: 11 saddr: 13 this node is: 14

uid 11 sender: 14 at: 0.364259

time: 0.373960 first: 0 seqno: 11 psize: 0 name: cbr direction: 1 npkt: 10 uid: 8 saddr: 13 this node is: 15

uid 8 sender: 15 at: 0.373985

==

congradulation! ack is received in:14

==

time: 0.387799 first: 0 seqno: 11 psize: 0 name: cbr direction: 1 npkt: 10 uid: 8 saddr: 13 this node is: 19

==

congradulation! ack is received in:13

==

time: 0.399577 first: 0 seqno: 12 psize: 0 name: cbr direction: 1 npkt: 11 uid: 10 saddr: 13 this node is: 15

uid 10 sender: 15 at: 0.399602

==

congradulation! ack is received in:15

==

time: 0.411240 first: 0 seqno: 12 psize: 0 name: cbr direction: 1 npkt: 11 uid: 10 saddr: 13 this node is: 19

time: 0.421311 first: 0 seqno: 13 psize: 0 name: cbr direction: 1 npkt: 12 uid: 11 saddr: 13 this node is: 15

uid 11 sender: 15 at: 0.421336

time: 0.431497 first: 0 seqno: 13 psize: 0 name: cbr direction: 1 npkt: 12 uid: 11 saddr: 13 this node is: 19

end simulation