

P-3574



**POWER MANAGEMENT IN WIRELESS
SENSOR NETWORKS THROUGH
SCHEDULING PROTOCOL**



PROJECT REPORT

Submitted by

G.RAJKUMAR

Reg. No: 0920108018

*In partial fulfillment for the award of the degree
of*

MASTER OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE - 641049

Department of Computer Science and Engineering

PROJECT WORK

APRIL 2011

This is to certify that the project entitled

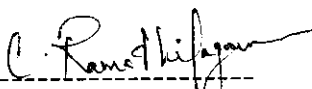
POWER MANAGEMENT IN WIRELESS SENSOR NETWORKS THROUGH SCHEDULING PROTOCOL

is the bonafide record of project work done by

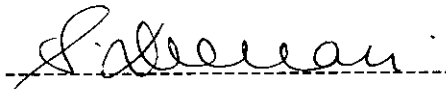
G.RAJKUMAR

Register No: 0920108018

of M.E. (Computer Science and Engineering) during the year 2010-2011.

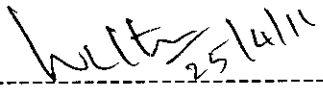


Project Guide




Head of the Department

Submitted for the Project Viva-Voce examination held on 25-04-2011



Internal Examiner

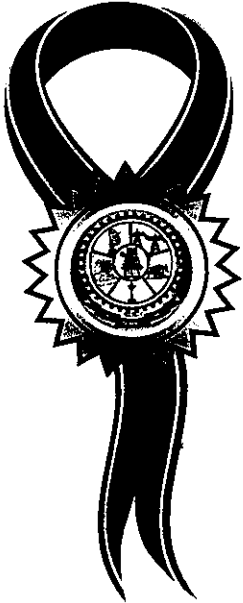


External Examiner 25/4/2011



MAHA COLLEGE OF ENGINEERING

Approved by AICTE, Affiliated to Anna University of Technology, Coimbatore.
Salem - Chennai Highway, Minnampalli, Salem - 636 106.



This is certified that Mr. / Ms. / Mrs. **G. RATKUMAR**.....
has participated and presented a paper entitled **POWER MANAGEMENT IN
WSN THROUGH SCHEDULING PROTOCOL**..... in the
National Conference on Emerging trends in Electronics & Communication
Engineering on 1-4-2011 in Maha College of Engineering, Salem.


Convener


Principal

DECLARATION

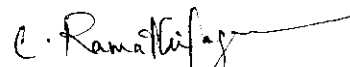
I affirm that the project work titled “**POWER MANAGEMENT IN WIRELESS SENSOR NETWORKS THROUGH SCHEDULING PROTOCOL**” being submitted in partial fulfillment for the award of **M.E (CSE)** degree is the original work carried out by me. It has not formed the part of any other project work submitted for the award of any degree or diploma, either in this or any other University.



G.RAJKUMAR

Register No: 0920108018

I certify that the declaration made above by the candidate is true to the best of my knowledge.



Mrs.C.RAMATHILAGAM, M.E.,

Assistant Professor

Department of Computer Science and Engineering,
Kumaraguru College of Technology,
(An Autonomous Institution)
Coimbatore-641 049.

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for enabling me to complete this project.

I express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E.**, for given this opportunity to pursue this course.

I would like to thank **Dr.S.Ramachandran, Ph.D.**, *Principal* for providing the necessary facilities to complete my thesis.

I thank **Mrs.S.Devaki, M.E.**, *Head of the Department*, Department of Computer Science and Engineering, for her support and timely motivation.

I thank all project committee members for their comments and advice during the reviews. Special thanks to **Mrs.V.Vanitha, M.E.**, *Assistant Professor* and **Mr.V.Subramani, M.Tech.**, *Associate Professor*, Department of Computer Science and Engineering, for arranging brain storming project review sessions.

I register my hearty appreciation to my Guide **Mrs.C.Ramathilagam, M.E.**, *Assistant Professor*, Department of Computer Science and Engineering, my project advisor. I thank for her support, encouragement and ideas. I thank her for the countless hours she has spent with me, discussing everything from research to academic choices.

I would like to convey my honest thanks to all **Teaching** staff and **Non Teaching** staffs of the department for their support. I would like to thank all my classmates who gave me a proper lighter moments and study breaks apart from extending some technical support whenever I needed them most.

I dedicate this project work to my **parents** for no reasons but feeling from bottom of my heart, without their love this work wouldn't be possible.

ABSTRACT

A wireless sensor network consists of a number of tiny sensor nodes deployed over a geographical area. Each node is a low-power device that integrates computing, wireless communication, and sensing capabilities. Sensor nodes are thus able to sense physical environmental information (e.g., temperature, pressure, vibrations) and process the acquired data locally, or send them to one or more collection points, usually referred to as sinks or base stations. A major problem for many sensor network applications is determining the most efficient way of conserving the energy of the power source. Regardless of the powering method, energy conservation is of prime importance for sensor networks. Energy saving is one critical issue for sensor networks since most sensors are equipped with non rechargeable batteries that have limited lifetime. Sensor nodes are generally powered by batteries which provide a limited lifetime and, often, cannot be replaced nor recharged, due to environmental or cost constraints. This project proposes scheduling protocol for efficient power management in wireless sensor networks targeted to periodic data acquisition. This protocol dynamically adjusts the sleep schedules of nodes to match the network demands and also it does not require any prior knowledge of the network topology or traffic pattern.

ஆய்வுச்சுருக்கம்

கம்பியில்லா உணர்வி வலையமைவானது, பல உணர்வி முனைப்புகளை உள்ளடக்கியது மற்றும் அவை நிலவியற் பிரிவில் செயற்படுத்தப்படுகிறது. ஒவ்வொரு முனைப்பும் குறைவான சக்தி கொண்டதாகவும் மற்றும் கணினி முறையையும், கம்பியில்லா தொடர்பு, உணர் திறமையை ஒன்றினைத்து உள்ளது. உணர்வி முனைப்புகள் பொருள் சுற்றுபுறத்தினை, (வெப்பம், அழுத்தம்) உணரக்கூடிய தன்மை கொண்டது. இத்தகவல் அனைத்தும் உணர்வி முனைப்புகளால் பெறப்பட்டு, தள நிலையத்திற்கு அனுப்பப்படுகிறது. கம்பியில்லா உணர்வி வலையத்தில், முக்கிய பயன்பாட்டில் ஒன்று திறமைமிக்க முறையை பயன்படுத்தி ஆற்றல் சிதைவடைதலை குறைப்பது. ஏனென்றால் உணர்வி முனைப்பின் மின்கலத்தில் குறைவான சக்தியே நிரப்பப்பட்டுள்ளது மற்றும் மறு மின்னூட்டு இல்லாததாகவும் உள்ளது. மின்கலத்தை எளிதாக மற்றும் தேவைகேற்ப மாற்ற இயலாது. இத்திட்டபணியில் அட்டவணை உடன்படு நெறிமுறை உருவாக்கப்பட்டு அதன் மூலம் ஆற்றல் இழத்தலை குறைக்கப்பட்டுள்ளது. இந்த அட்டவணை உடன்படு நெறிமுறை மாறும் நிலை ஆய்வுக்கு உடன்படுவதாகவும், பிணையத்தின் கோருதல்களை நிறைவேற்ற கூடியதாகவும் உருவாக்கப்பட்டுள்ளது. அட்டவணை உடன்படு நெறிமுறை கொண்டு எளிதாக ஆற்றல் சிதைவடைதலை சரி செய்யலாம் மற்றும் கம்பியில்லா உணர்வி வலையத்தின் ஆயுட்காலத்தை அதிகரிக்கலாம்.

TABLE OF CONTENTS

CONTENTS	PAGE NO
ABSTRACT	v
ABSTRACT (TAMIL)	vi
LIST OF FIGURES	x
LIST OF ABBREVIATION	xi
LIST OF SYMBOLS	xii
1. INTRODUCTION	
1.1 Overview of Wireless Sensor Networks	1
1.2 Sensor Node System Architecture	4
1.3 Components of Sensor Networks	5
1.4 Applications of Sensor Networks	7
1.5 Challenges of Wireless Sensor Networks	8
2. LITERATURE REVIEW	
2.1 Power Management in Wireless Sensor Networks	11
2.2 Need for Power Management in Wireless Sensor Networks	11
2.3 Power Saving Modes in Sensors	14
2.4 Existing System	17
2.5 Related Works	20
3. SYSTEM REQUIREMENTS	
3.1 Hardware Requirements	22
3.2 Software Requirements	22

4. SOFTWARE DESCRIPTION

4.1 Network Simulator	23
4.2 User's View of Ns-2	24
4.3 Network Components	24
4.4 Class Tcl	25
4.4.1 Obtain a Reference to the class Tcl instance	26
4.4.2 Invoking OTcl Procedures	26
4.5 Command Methods: Definition and Invocation	26
4.6 Trace Analysis	26
4.7 Network Animator (NAM)	27

5. PROJECT DESCRIPTION

5.1 Protocol Definition	29
5.2 Overview of the Project	29
5.3 Module Description	31
5.3.1 <i>Talk Interval Prediction</i>	31
5.3.2 <i>Sleep Coordination</i>	32
5.3.3 <i>Schedule Robustness</i>	33
5.3.3.1 <i>Beacon Protection</i>	33

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Simulation Scenario	38
6.2 Performance Evaluation	38
6.3 Performance Comparison	39
6.3.1 AODV Protocol	39
6.3.2 Scheduling Protocol	39

7. CONCLUSION AND FUTURE OUTLOOK

7.1 Conclusion 41

7.2 Future Outlook 41

8. APPENDIX

8.1 Sample Source Code 42

8.2 Snapshots 54

9. REFERENCES

62

LIST OF FIGURES

FIGURE NO	CAPTION	PAGE NO
1.1	Wireless Sensor Networks	2
1.2	Function of Wireless Sensor Network	3
1.3	Sensor Node System Architecture	4
2.1	Power Saving Modes in WSN	14
2.2	Routing Protocols	17
4.1	Block diagram of Architecture of NS-2	24
4.2	OTcl Class Hierarchy	25
4.3	Trace format example	27
4.4	Screenshot of NAM window	28
5.1	Parameters of the sleep scheduling protocol	30
6.1	Comparison of AODV and Scheduling Protocol	40

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
WSN	Wireless Sensor Network
ADC	Analog to Digital Converter
RF	Radio Frequency
CDMA	Code Division Multiple Access
SIR	Signal to Interference Ratio
DSDV	Destination-Sequence Distance Vector
RREQ	Route Request
RREP	Route Reply
AODV	Ad hoc On demand Distance Vector routing
NS	Network Simulator
NAM	Network Animator
MAC	Medium Access Control

LIST OF SYMBOLS

SYMBOLS	MEANING
m	Message inter-reception time
n_{pkt}	Number of received messages
CP	length of the communication period
$t_{parent,j}^{(m+1)}$	Next wakeup time of parent node
TI	length of the next talk interval
q	Time Slot

CHAPTER 1

INTRODUCTION

1.1 WIRELESS SENSOR NETWORKS

A WSN can be defined as a network of devices, denoted as nodes, which can sense the environment and communicate the information gathered from the monitored field (e.g., an area or volume) through wireless links. The data is forwarded, possibly via multiple hops, to a sink (sometimes denoted as controller or monitor) that can use it locally or is connected to other networks (e.g., the Internet) through a gateway. The nodes can be stationary or moving.

Sensor networks are highly distributed networks of small, lightweight wireless nodes, deployed in large numbers to monitor the environment or system by the measurement of physical parameters such as temperature, pressure, or relative humidity. Wireless Sensor Networks have the potential to revolutionize sensing technology. The large number ensures that at least some of the sensors will be close to the phenomenon of interest and thus be able to have high quality measurements. In-network processing allows tracking of targets and the evolution of the studied phenomena. It also allows for substantial power savings and reduced bandwidth necessary to observe certain phenomena. The large number of sensors also increases the reliability of the system, as failure of a percentage of the sensor nodes will not result in system failure. Like any other electronic devices, sensor nodes have to be powered. In most applications, a power cable is not an option. A very popular method of powering wireless sensor networks is with the use of batteries. Sensor nodes are expected to have very small form factors. This precludes spending a large amount of resources on a large, expensive power source for each node. Once deployed, the sensor networks are usually unattended. The life of the sensor network is determined by the life of its batteries. Finally, such a network is typically expected to work for extended periods of time (weeks, months, and in some cases years).

Network lifetime has become the key characteristic for evaluating sensor networks in an application-specific way. Especially the availability of nodes, the sensor coverage, and the connectivity have been included in discussions on network lifetime. Even quality of service measures can be reduced to lifetime considerations.

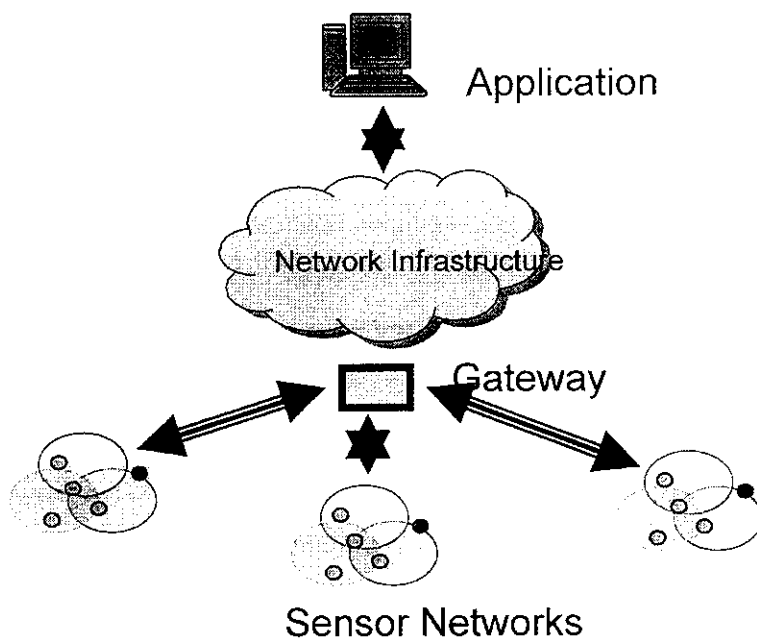


Figure 1.1 Wireless Sensor Networks

A great number of algorithms and methods were proposed to increase the lifetime of a sensor network while the evaluations were always based on a particular definition of network lifetime. Network lifetime strongly depends on the lifetimes of the single nodes that constitute the network. This fact does not depend on how the network lifetime is defined.

Thus, if the lifetimes of single nodes are not predicted accurately, it is possible that the derived network life time metric will deviate in an uncontrollable manner. It should therefore be clear that accurate and consistent modeling of the single nodes is very important.

The lifetime of a sensor node basically depends on two factors: how much energy it consumes over time, and how much energy is available for its use. Naturally, lifetime was then discussed from different points of view, which led to the development of various lifetime metrics. Depending on the energy consumers regarded in each metric and the specific application requirements considered, these metrics may lead to very different estimations of network lifetime. Building sensors has been made possible by the recent advances in micro-electro mechanical systems (MEMS) technology.

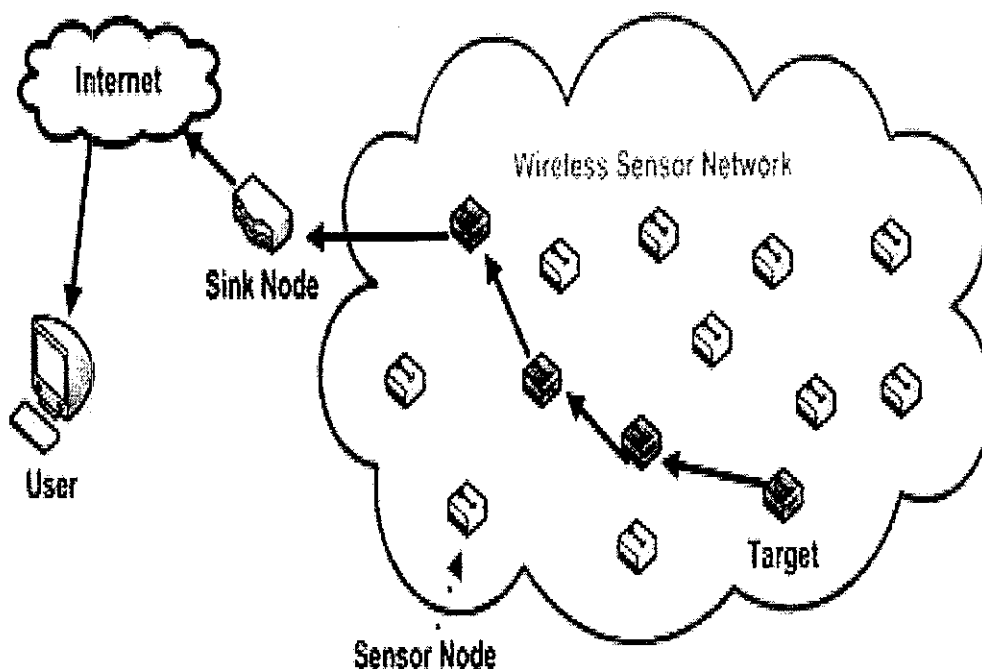


Figure 1.2 Function of Wireless Sensor Network

Single-sink [Fig1.2] scenario suffers from the lack of scalability by increasing the number of nodes, the amount of data gathered by the sink increases and once its capacity is reached, the network size cannot be augmented. Moreover, for reasons related to MAC and routing aspects, network performance cannot be considered independent from the network size.

In principle, a multiple-sink WSN can be scalable (i.e., the same performance can be achieved even by increasing the number of nodes), while this is clearly not true for a single-sink network. In many cases nodes send the data collected to one of the sinks, selected among many, which forward the data to the gateway, toward the final user.

From the protocol viewpoint, this means that a selection can be done, based on a suitable criteria that could be, for example, Minimum delay, Maximum throughput, minimum number of hops, etc. Therefore, the presence of multiple sinks ensures better network performance with respect to the single-sink case (assuming the same number of nodes is deployed over the same area), but the communication protocols must be more complex and should be designed according to suitable criteria.

Each node of the sensor network consists of three subsystems: the sensor sub system which senses the environment, the processing subsystem which performs local computations on the sensed data, and the communication subsystem which is responsible for message exchange with neighboring sensor nodes. While individual sensors have limited sensing region , processing power, and energy, networking a large number of sensors gives rise to a robust, reliable, and accurate sensor network covering a wider region.

1.2 SENSOR NODE SYSTEM ARCHITECTURE

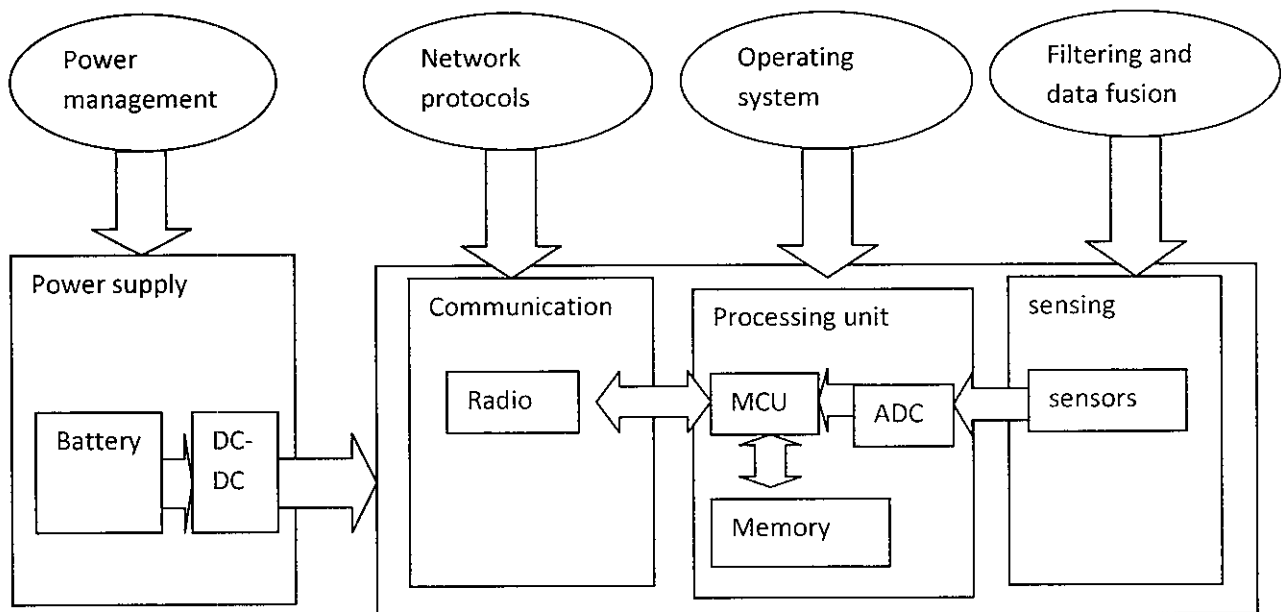


Figure 1.3 Sensor Node System Architecture

This system architecture of a generic sensor node is composed of four major blocks: power supply, communication, processing unit and sensors.

- The power supply block consists of a battery and the dc-dc converter and has the purpose to power the node.
- The communication block consists of a wireless communication channel. Most of the platforms use short range radio. Others solutions includes laser and infrared. The processing unit is composed of memory to store data and applications programs,

a microcontroller and an Analog-to-Digital Converter to receive signal from the sensing block.

- This last block links the sensor node to the physical world and has a group of sensors and actuators that depends on the application of the wireless sensor network.

1.3 COMPONENTS OF SENSOR NETWORKS

Microcontroller

Microcontroller performs tasks, processes data and controls the functionality of other components in the sensor node. Other alternatives that can be used as a controller are: General purpose desktop microprocessor, Digital signal processors, Field Programmable Gate Array and Application-specific integrated circuit. Microcontrollers are the most suitable choice for a sensor node.

Transceiver

Sensor nodes make use of ISM band which gives free radio, huge spectrum allocation and global availability. The various choices of wireless transmission media are Radio frequency, Optical communication (Laser) and Infrared. Laser requires less energy, but needs line-of-sight for communication and also sensitive to atmospheric conditions. Infrared like laser, needs no antenna but is limited in its broadcasting capacity. Radio Frequency (RF) based communication is the most relevant that fits to most of the WSN applications. WSN's use the communication frequencies between about 433 MHz and 2.4 GHz. The functionality of both transmitter and receiver are combined into a single device known as transceivers are used in sensor nodes. Transceivers lack unique identifier. The operational states are Transmit, Receive, Idle and Sleep.

External Memory

From an energy perspective, the most relevant kinds of memory are on-chip memory of a microcontroller and Flash memory - off-chip RAM is rarely if ever used. Flash memories are used due to its cost and storage capacity. Memory requirements are very much application dependent.

Two categories of memory based on the purpose of storage a) User memory used for storing application related or personal data. b) Program memory used for programming the device. This memory also contains identification data of the device if any.

Sensors

Sensors are hardware devices that produce measurable response to a change in a physical condition like temperature and pressure.

Sensors sense or measure physical data of the area to be monitored. The continual analog signal sensed by the sensors is digitized by an Analog-to-digital converter and sent to controllers for further processing.

Characteristics and requirements of Sensor node should be small size, consume extremely low energy, operate in high volumetric densities, be autonomous and operate unattended, and be adaptive to the environment. As wireless sensor nodes are micro-electronic sensor device, can only be equipped with a limited power source of less than 0.5 Ah and 1.2 V.

ADC - Analog to Digital Converter

The analog signals produced by the sensors are converted to digital signals by the analog to digital converter (ADC), and then fed into the processing unit.

Power Source

Power consumption in the sensor node is for the Sensing, Communication and Data Processing. More energy is required for data communication in sensor node. Energy expenditure is less for sensing and data processing. The energy cost of transmitting 1 Kb a distance of 100 m is approximately the same as that for the executing 3 million instructions by 100 million instructions per second/W processor. Power is stored either in Batteries or Capacitors. Batteries are the main source of power supply for sensor nodes.

A power management layer is to control the main resource of a sensor node, its energy level. The power management layer could use the knowledge of battery's voltage slope to adapt dynamically the system performance. Another advantage is that other energy source can be added and the power management can make the best use of the energy resource.

The sensing unit is composed of a group of sensor, which are devices that produce an electrical response to a change in a physical condition.

The type of sensors being used in a sensor node will depend on the application. Processing unit since the sensor node is expected to communicate, to process and to gather sensor data, sensor nodes must have processing units. The central processing unit of a sensor node determines to a large degree both the energy consumption as well as the computational capabilities of a sensor node.

1.4 APPLICATIONS OF SENSOR NETWORKS

Wireless Sensor Networks have a wide range of applications such as,

1. Military applications
 - i. Monitoring friendly forces and equipment.
 - ii. Battlefield surveillance.
 - iii. Nuclear, biological and chemical attack detection.
2. Environmental Applications
 - i. Forest fire detection.
 - ii. Bio-complexity mapping of the environment.
 - iii. Flood detection.
3. Health applications
 - i. Tele-monitoring of human physiological data.
 - ii. Tracking and monitoring doctors and patients inside a hospital.
 - iii. Drug administration in hospitals.
4. Home application
 - i. Home automation.
 - ii. Smart environment

In order to enable reliable and efficient observation and initiate right actions, physical phenomenon features should be reliably detected/estimated from the collective information provided by sensor nodes. Moreover, instead of sending the raw data to the nodes responsible for the fusion, sensor nodes use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data. Hence, these properties of WSN impose unique challenges for development of communication protocols in such architecture.

1.5 CHALLENGES OF WIRELESS SENSOR NETWORKS

- **Energy efficiency/system lifetime**

As sensor nodes are battery-operated, protocols must be energy-efficient to maximize system life time. System life time can be measured such as the time until half of the nodes die or by application-directed metrics, such as when the network stops providing the application with the desired information about the phenomena.

- **Fault Tolerance**

Some sensor nodes may fail or be blocked due to lack of power, have physical damage or environmental interference. The failure of sensor nodes should not affect the overall task of the sensor network. This is the reliability or fault tolerance issue. Fault tolerance is the ability to sustain sensor network functionalities without any interruption due to sensor node failures.

- **Scalability**

The number of sensor nodes deployed in studying a phenomenon may be in the order of hundreds or thousands. Depending on the application, the number may reach an extreme value of millions. The new schemes must be able to work with this number of nodes.

- **Production Costs**

Since the sensor networks consist of a large number of sensor nodes, the cost of a single node is very important to justify the overall cost of the networks. If the cost of the network is more expensive than deploying traditional sensors, then the sensor network is not cost-justified. As a result, the cost of each sensor node has to be kept low.

- **Environment**

Sensor nodes are densely deployed either very close or directly inside the phenomenon to be observed. Therefore, they usually work unattended in remote geographic areas. They may be working in busy intersections, in the interior of large machinery, at the bottom of an ocean, inside a twister, on the surface of an ocean. They work under high pressure in the bottom of an ocean, in harsh environments such as debris or a battlefield,

under extreme heat and cold such as in the nozzle of an aircraft engine or in arctic regions, and in an extremely noisy environment such as under intentional jamming.

- **Hardware Constraints**

A sensor node is made up of four basic components: a sensing unit, a processing unit, a transceiver unit and a power unit. Sensing units are usually composed of two subunits: sensors and analog to digital converters (ADCs). The analog signals produced by the sensors based on the observed phenomenon are converted to digital signals by the ADC, and then fed into the processing unit. The processing unit, which is generally associated with a small storage unit, manages the procedures that enable the sensor node collaborate with the other nodes to carry out the assigned sensing tasks. A transceiver unit connects the node to the network. One of the most important components of a sensor node is the power unit. Power units may be supported by a power scavenging unit such as solar cells. There are also other subunits, which are application dependent. Most of the sensor network routing techniques and sensing tasks require the knowledge of location with high accuracy.

- **Sensor Network Topology**

Sheer numbers of inaccessible and unattended sensor nodes, which are prone to frequent failures, make topology maintenance a challenging task. Hundreds to several thousands of nodes are deployed throughout the sensor field.

- a) **Pre-Deployment Phase**

Sensor nodes can be either thrown in mass or placed one by one in the sensor field. They can be deployed by dropping from a plane, delivering in an artillery shell, rocket or missile, throwing by a catapult, placing in factory, and placing one by one either by a human or a robot. Although the sheer number of sensors and their unattended deployment usually preclude placing them according to a carefully engineered deployment plan, the schemes for initial deployment must reduce the installation cost, eliminate the need for any pre-organization and preplanning, increase the flexibility of arrangement, and promote self-organization and fault tolerance.

b) Post-Deployment Phase

After deployment, topology changes are due to change in sensor nodes position, reachability (due to jamming, noise, moving obstacles, etc.), available energy, malfunctioning, and task details. Sensor nodes may be statically deployed. However, device failure is a regular or common event due to energy depletion or destruction.

It is also possible to have sensor networks with highly mobile nodes. Besides, sensor nodes and the network experience varying task dynamics, and they may be a target for deliberate jamming. Therefore, sensor network topologies are prone to frequent changes after deployment.

c) Re-Deployment of Additional Nodes Phase

Additional sensor nodes can be re-deployed at any time to replace the malfunctioning nodes or due to changes in task dynamics. Addition of new nodes poses a need to re-organize the network. Coping with frequent topology changes in an ad hoc network that has myriads of nodes and very stringent power consumption constraints requires special routing protocols.

- **Transmission Media**

In a multi-hop sensor network, communicating nodes are linked by a wireless medium. These links can be formed by radio, infrared or optical media. To enable global operation of these networks, the chosen transmission medium must be available worldwide.

- **Power Consumption**

The wireless sensor node, being a microelectronic device, can only be equipped with a limited power source. In some application scenarios, replenishment of power resources might be impossible. Sensor node lifetime, therefore, shows a strong dependence on battery lifetime. The malfunctioning of few nodes can cause significant topological changes and might require rerouting of packets and re-organization of the network. In sensor networks, power efficiency is an important performance metric, directly influencing the network lifetime. Application specific protocols can be designed by appropriately trading off other performance metrics such as delay and throughput with power efficiency. The main task of a sensor node in a sensor field is to detect events, perform quick local data processing, and then transmit the data. Power consumption can hence be divided into three domains: sensing, communication, and data processing.

CHAPTER 2

LITERATURE REVIEW



2.1 POWER MANAGEMENT IN WIRELESS SENSOR NETWORKS

The nodes in wireless sensor network are constrained by limited battery power for their operation. Hence, energy management is an important issue in such networks. The use of multi-hop radio relaying requires a sufficient number of precious resources that must be used efficiently in order to avoid early termination of any node.

Energy management deals with the process of managing energy resources by means of controlling the battery discharge, adjusting the transmission power, and scheduling of power sources so as to increase the lifetime of the nodes of wireless sensor network. Efficient battery management, transmission power management and system power management are three major means of increasing the life of a node.

- Battery power management is concerned with problems that lie in the selection of battery technologies, finding the optimal capacity of the battery, and scheduling of batteries , that increase the battery capacity.
- Transmission power management techniques attempt to find an optimum power level for the nodes in the ad hoc wireless network.
- System power management deals mainly with minimizing the power required by hardware peripherals of a node and in cooperating low-power strategies into the protocols used in various layers of the protocol stack.

2.2 NEED FOR POWER MANAGEMENT IN WIRELESS SENSOR NETWORKS

The energy efficiency of a node is defined as the ratio of the amount of data delivered by the node to the total energy expended. Higher energy efficiency implies that a greater number of packets can be transmitted by the node with a given amount of energy source.

The main reasons for energy management in wireless sensor networks are listed below:

Limited energy source

Advances in battery technologies have been negligible as compared to the recent advances that have taken place in the field of mobile computing and communication. The increasing gap between the power consumption requirements and power availability adds to the importance of energy management.

Difficulties in replacing the batteries

Sometimes it become very difficult to replace or recharge the batteries. in situations such as battlefields, this is almost impossible. Hence, energy conservation is essential in such scenarios.

Constraints on the battery source

Batteries tend to increase the size and weight of a mobile node. Reducing the size of the battery results in less capacity which, in turn, decreases the active lifespan of the node. Hence, in addition to reducing the size of the battery, energy management techniques are necessary to utilize the battery capacity in the best possible way.

Selection of optimal transmission power

The transmission power selected determines the reach ability of the nodes. The consumption of battery charge increase with an increase in the transmission power. An optimal value for the transmission power decreases the interference among nodes, which, in turn, increases the number of simultaneous transmissions.

Channel utilization

A reduction in the transmission power increases frequency reuse, which leads to better channel reuse. Power control becomes very important for CDMA-based systems in which the available bandwidth is shared among all the users. Hence the power control is essential to maintain the required signal to interference ratio (SIR) at the receiver and to increase the channel reusability.

Many devices such as Mica2 and MicaZ that are used in WSN run on two AA batteries. Depending on the activity level of a node, its lifetime may only be a few days if no power management schemes are used. Since most systems require much longer lifetime, significant research has been undertaken to increase lifetime while still meeting functional requirements.

At the hardware level it is possible to add solar cells or scavenge energy from motion or wind. Batteries are also improving. If form factor is not a problem then it is also possible to add even more batteries. Low power circuits and microcontrollers are improving. Most hardware platforms allow multiple power saving states (off, idle, on) for each component of the device (each sensor, the radio, the microcontroller). In this way, only the components required at a particular time need to be active.

At the software level power management solutions are targeted at (i) minimizing communications since transmitting and listening for messages is energy expensive, and (ii) creating sleep/wake-up schedules for nodes or particular components of nodes.

Minimizing the number of messages is a cross-cutting problem. For example, with a good MAC protocol there are fewer collisions and retries. With good routing, short paths and congestion avoidance or minimization can be achieved and this minimizes the number of messages sent. Efficient neighbor discovery, time synchronization, localization, query dissemination and flooding can all reduce the number of messages thereby increasing lifetime. Solutions to schedule sleep/wake-up patterns vary considerably. Many solutions attempt to keep awake the minimum number of nodes, called sentries, to provide the required sensing coverage while permitting all the others to sleep. To balance energy consumption a rotation is performed periodically where new sentries are selected for the next period of time. Another common technique is to duty-cycle nodes.

2.3 POWER SAVING MODES IN SENSORS

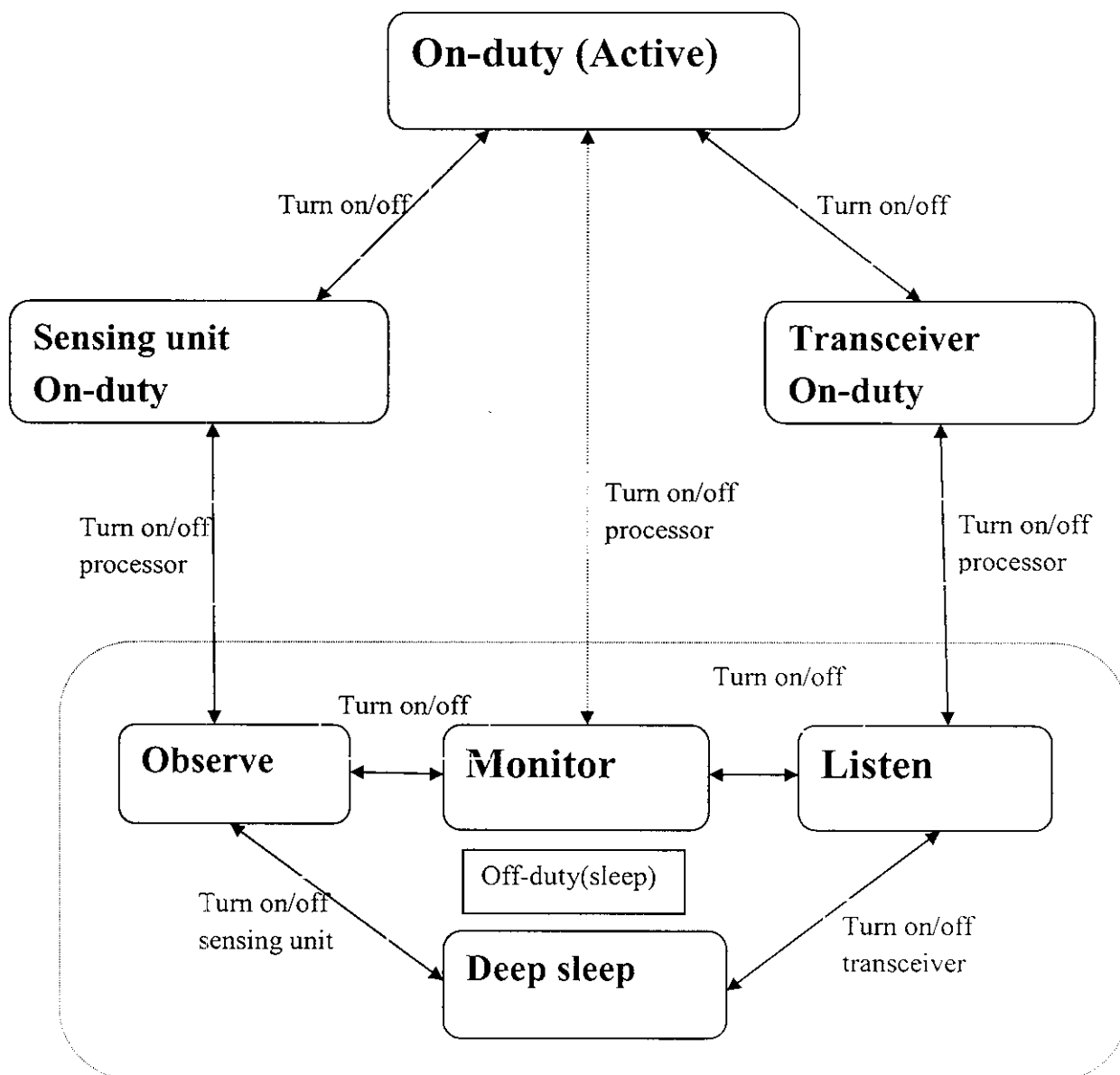


Figure 2.1 Power Saving Modes in Wireless Sensor Networks

In order to make the sensor mechanisms, one first needs to be able to differentiate the various energy saving modes that can be provided by a sensor. One complexity here is that different types of sensors may support different sets of modes and even if they support the same set of modes, they often use different terminology.

The major modes of a sensor as follows:

On-Duty

All the components in the sensor are turned on. The sensor is able to collect sensory data, send/receive messages, process data and messages, and do other types of computation. This mode is also called the active mode in the literature. It is not an energy-saving mode.

Sensing Unit On-Duty

At least one sensing unit and the processor are turned on, but the transceiver is turned off. In this mode, the sensor is capable of sensing and processing sensory data, but not transmitting or receiving messages.

Transceiver On-Duty

The transceiver and the processor are turned on, but all the sensing units are turned off. In this mode, the sensor is capable of transmitting, receiving and processing messages, but not sensing. And also use the shorter form TR On-Duty.

Off-Duty

The sensor's processor is turned off, but a timer or some other triggering mechanism may be running to wake up the sensor. This mode is also called the Sleep mode in the literature. Some sensors have multiple Off-Duty (Sleep) modes, each with a different wakeup mechanism.

For example, the mAMPS sensor has three sleep modes: Monitor, Observe, and Deep Sleep. The processor is turned off in all three modes, so the sensor cannot process any sensory data or messages. However, in the Monitor mode, both the sensing unit and the transceiver are left on to receive wakeup signals. In the Observe mode, only the sensing unit is on.

The Observe mode is different from the SU-On-Duty mode as the processor is turned on in the latter. In the Deep Sleep mode, neither the sensing unit nor the transceiver is turned on, so the sensor relies on a preset internal timer to wake itself up.

Most sensors provide a sleep mode similar to mAMPS' Deep Sleep mode. It is possible to design a fourth sleep mode that they call the Listen mode. In this mode, only the transceiver is turned on to receive wakeup signals (but the processor and sensing unit are off).

In the power savings modes in sensor show how a sensor can change from one mode to another. A distributed scheduling mechanism allows each sensor to determine when it should switch its mode and what mode it will switch to. Such a mechanism should minimize the overall energy consumption and achieve the application-specific objectives.

Sensors consume the most energy in the On-Duty mode and save the most energy in the Deep Sleep mode. The energy consumption in the other modes depends on the design of the sensors, but in general one can expect the following: the energy consumption of the Observe and Listen modes should be less than that of the Monitor mode which in turn should be less than that of the SU-On-Duty and TU-On-Duty modes. Moreover, sensors generally save more energy in the SU-On-Duty mode than in the TR-On-Duty mode, because communication usually consumes more energy than sensing. However, the energy consumption in the TR-On-Duty mode also depends on the actual communication pattern and frequency.

First, the energy cost of transmitting messages may be quite different from that of receiving messages. Normally, transmitting messages costs more energy than receiving messages, but there are exceptions. For example, the MICAz sensor draws 19.7 mA of current when receiving a message and 17 mA of current when transmitting at the 1 mW level (the highest transmission power).

Second, more frequent communication may consume more energy. Although idling may consume a similar level of energy as receiving (or even sending) a message, this may not apply to all sensors.

Moreover, frequent communication can cause collisions and retransmissions, which will lead to more energy consumption. Almost all of the surveyed mechanisms take advantage of the energy saving feature of the Deep Sleep mode. However, the designers of mAMPS proposed a mechanism to take advantage of all of its three sleeping modes.

2.4 EXISTING SYSTEM

Table-Driven (or Proactive)

The nodes maintain a table of routes to every destination in the network, for this reason they periodically exchange messages. At all times the routes to all destinations are ready to use and as a consequence initial delays before sending data are small. Keeping routes to all destinations up-to-date, even if they are not used, is a disadvantage with regard to the usage of bandwidth and of network resources.

On-Demand (or Reactive)

These protocols were designed to overcome the wasted effort in maintaining unused routes. Routing information is acquired only when there is a need for it. The needed routes are calculated on demand. This saves the overhead of maintaining unused routes at each node, but on the other hand the latency for sending data packets will considerably increase.

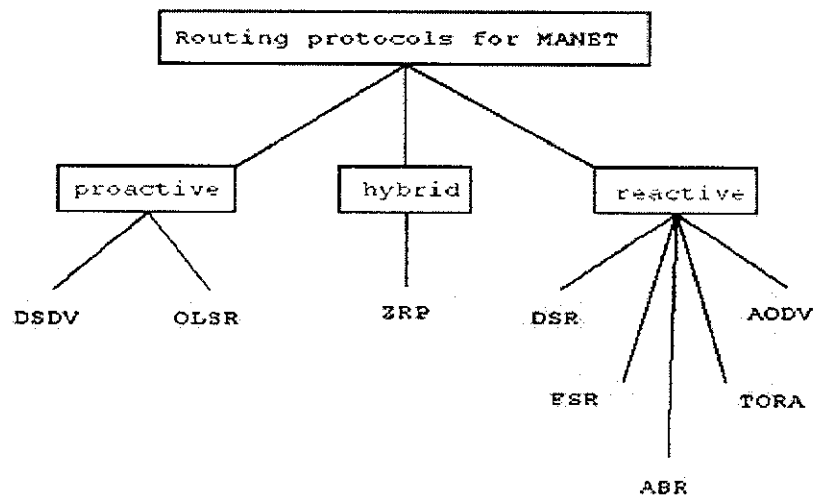


Figure 2.2 Routing Protocols

PROACTIVE:**DSDV (Destination-Sequence Distance Vector)**

DSDV has one routing table, each entry in the table contains: destination address, number of hops toward destination, next hop address. Routing table contains all the destinations that one node can communicate. When a source A communicates with a destination B, it looks up routing table for the entry which contains destination address as B. Next hop address C was taken from that entry. A then sends its packets to C and asks C to forward to B. C and other intermediate nodes will work in a similar way until the packets reach B. DSDV marks each entry by sequence number to distinguish between old and new route for preventing loop.

DSDV use two types of packet to transfer routing information: full dump and incremental packet. The first time two DSDV nodes meet, they exchange all of their available routing information in full dump packet. From that time, they only use incremental packets to notice about change in the routing table to reduce the packet size. Every node in DSDV has to send update routing information periodically. When two routes are discovered, route with larger sequence number will be chosen. If two routes have the same sequence number, route with smaller hop count to destination will be chosen.

DSDV has advantages of simple routing table format, simple routing operation and guarantee loop-freedom.

The disadvantages are,

- a large overhead caused by periodical update
- Waste resource for finding all possible routes between each pair, but only one route is used.

REACTIVE:**On-demand Routing Protocols**

In on-demand trend, routing information is only created to requested destination. Link is also monitored by periodical Hello messages.

If a link in the path is broken, the source needs to rediscover the path. On-demand strategy causes less overhead and easier to scalability. However, there is more delay because the path is not always ready.

AODV Routing

Ad hoc on demand distance vector routing (AODV) is the combination of DSDV and DSR.

In AODV, each node maintains one routing table. Each routing table entry contains:

- Active neighbor list: a list of neighbor nodes that are actively using this route entry.
- Once the link in the entry is broken, neighbor nodes in this list will be informed.
- Destination address.
- Next-hop address toward that destination.
- Number of hops to destination.
- Sequence number: for choosing route and prevent loop.
- Lifetime: time when that entry expires.

Routing in AODV consists of two phases: Route Discovery and Route Maintenance. When a node wants to communicate with a destination, it looks up in the routing table. If the destination is found, node transmits data in the same way as in DSDV. If not, it start Route Discovery mechanism: Source node broadcast the Route Request packet to its neighbor nodes, which in turns rebroadcast this request to their neighbor nodes until finding possible way to the destination. When intermediate node receives a RREQ, it updates the route to previous node and checks whether it satisfies the two conditions:

- (i) There is an available entry which has the same destination with RREQ
- (ii) Its sequence number is greater or equal to sequence number of RREQ. If no, it rebroadcast RREQ. If yes, it generates a RREP message to the source node.

When RREP is routed back, node in the reverse path updates their routing table with the added next hop information. If a node receives a RREQ that it has seen before (checked by the sequence number), it discards the RREQ for preventing loop. If source node receives more than one RREP, the one with greater sequence number will be chosen. For two RREPs with the same sequence number, the one with less number of hops to destination will be chosen. When a route is found, it is maintained by Route Maintenance mechanism: Each node periodically send Hello packet to its neighbors for proving its availability. When Hello packet is not received from a node in a time, link to that node is considered to be broken. The node which does not receive Hello message will invalidate all of its related routes to the failed node and inform other neighbor using this node by Route Error packet. The source if still want to transmit data to the destination should restart Route Discovery to get a new path. AODV has advantages of decreasing the overhead control messages, low processing, quick adapt to net work topology change, more scalable up to 10000 mobile nodes. However, the disadvantages are that AODV only accepts bi-directional link and has much delay when it initiates a route and repairs the broken link.

2.5 RELATED WORKS

In the literature, can find a great number of relevant publications that address the problem of sensor network lifetime. Some papers employ network lifetime as a criterion that needs to be maximized, but never exactly define the term network lifetime.

Network Lifetime Based on Sensor Coverage

Considering the specific characteristics of sensor networks, measuring the network lifetime as the time that the region of interest is covered by sensor nodes seems to be a natural way to define the lifetime. Coverage can be defined in different ways, depending on the composition of the region of interest and the achieved redundancy of the coverage. The region of interest can be a two-dimensional area or a three-dimensional volume where each point inside the area or volume has to be covered. This is often referred to as area or volume coverage.

If only a finite set of target points inside an area has to be covered, the corresponding coverage problem is called target coverage.

A third coverage problem, barrier coverage, describes the chance that that a mobile target can pass undetected through a barrier of sensor nodes.

There are two approaches to describe the degree of coverage redundancy that can be achieved by a given sensor network. The first approach requires that only a given percentage α , of the region of interest, is covered by at least one sensor. This is commonly called α -coverage. The second approach aims to achieve more redundancy, and thus requires that each point within the region of interest is covered by at least k sensors. This is termed k -coverage. Sensor coverage is often argued to be the most important measure for the quality of service a sensor network provides. There is a lot of ongoing research concerning coverage in sensor networks, often in the context of deployment strategies or scheduling algorithms. However, defining network lifetime solely based on the achieved coverage is not sufficient for most application scenarios because it is not guaranteed that the measured data can ever be transmitted to a sink node.

Network Lifetime Based on Connectivity

Another group of metrics takes the connectivity of the network into account. Connectivity is a metric that is commonly encountered in the context of ad hoc networks because there is no notion of sensor coverage in ad hoc networks and thus the ability to transmit data to a given destination is most important. The lifetime as the minimum time when either the percentage of alive nodes or the size of the largest connected component of the network drop below a specified threshold. However, this definition only considers the size of the largest connected component in the network. This is clearly insufficient in WSNs where connectivity towards a base station is what matters most.

Integrating connectivity in a network lifetime metric is certainly a good idea. However, it is important to consider connectivity towards a base station, not just connections between arbitrary sensor nodes. In addition, measuring the lifetime of a connected network in terms of numbers of transmitted packets is not comparable across different networks, and gives no indication of the absolute network lifetime.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS:

Processor	: Pentium III
Processor speed	: 1.5 GHZ
Memory (RAM)	: 256MB
Hard disk	: 40GB
Monitor	: 15" color monitor
Keyboard	: Logitech 104 keys
Mouse	: Logitech scroll mouse

3.2 SOFTWARE REQUIREMENTS:

Operating System	: Fedora 8.0
Language	: C++ & TCL Scripting

CHAPTER 4

SOFTWARE DESCRIPTION

Network simulator-2 is used as the simulation tool in this project. NS was chosen as the simulator partly because of the range of features it provides and partly because it has an open source code that can be modified and extended.

4.1 NETWORK SIMULATOR (NS)

Network simulator (NS) is an object oriented, discrete event simulator for networking research. NS provides substantial support for simulation of TCP, routing and multicast protocols over wired and wireless networks. The simulator is a result of an ongoing effort of research and developed. Even though there is a considerable confidence in NS, it is not a polished product yet and bugs are being discovered and corrected continuously.

NS is written in C++, with an OTcl1 interpreter as a command and configuration interface. The C++ part, which is fast to run but slower to change, is used for detailed protocol implementation. The OTcl part, on the other hand, which runs much slower but can be changed very fast quickly, is used for simulation configuration. One of the advantages of this split-language program approach is that it allows for fast generation of large scenarios. To simply use the simulator, it is sufficient to know OTcl. On the other hand, one disadvantage is that modifying and extending the simulator requires programming and debugging in both languages.

NS can simulate the following:

- 1. Topology:** Wired, Wireless
- 2. Scheduling Algorithms:** RED, Drop Tail,
- 3. Transport Protocols:** TCP, UDP
- 4. Routing:** Static and Dynamic routing
- 5. Application:** FTP, HTTP, Telnet, Traffic generators

4.2 USER'S VIEW OF NS-2

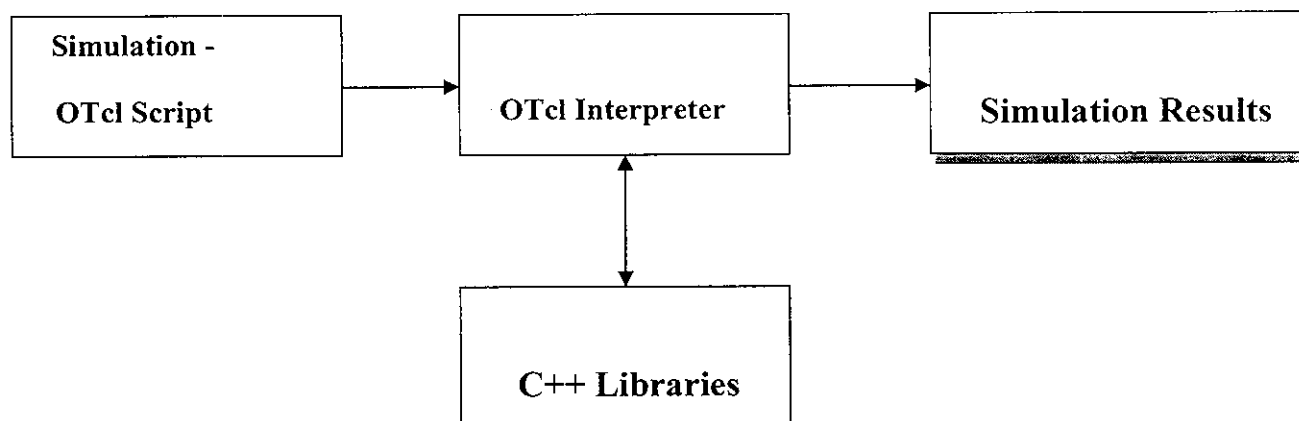


Figure 4.1 Block diagram of Architecture of NS-2

4.3 NETWORK COMPONENTS

This section talks about the NS components, mostly compound network components. Figure 4.2 shows a partial OTcl class hierarchy of NS, which will help understanding the basic network components.

The root of the hierarchy is the TclObject class that is the super class of all OTcl library objects (scheduler, network components, timers and the other objects including NAM related ones). As an ancestor class of TclObject, Ns Object class is the super class of all basic network component objects that handle packets, which may compose compound network objects such as nodes and links. The basic network components are further divided into two subclasses, Connector and Classifier, based on the number of the possible output DATA paths. The basic network and objects that have only one output DATA path are under the Connector class, and switching objects that have possible multiple output DATA paths are under the Classifier class.

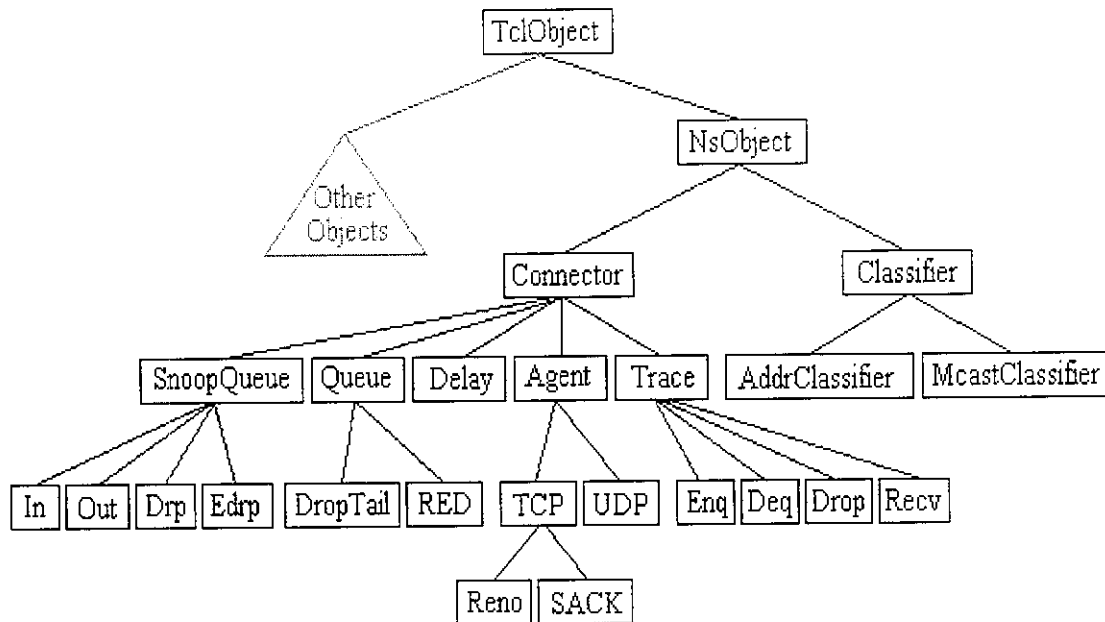


Figure 4.2 OTcl Class Hierarchy

4.4 CLASS TCL

The class Tcl encapsulates the actual instance of the OTcl interpreter and provides the methods to access and communicate with that interpreter, code. The class provides methods for the following operations:

1. Obtain a reference to the Tcl instance
2. Invoke OTcl procedures through the interpreter
3. Retrieve, or pass back results to the interpreter
4. Report error situations and exit in an uniform manner
5. Store and lookup "TclObjects"
6. Acquire direct access to the interpreter.

4.4.1 Obtain a Reference to the class Tcl instance

A single instance of the class is declared in `-tclcl/Tcl.cc` as a static member variable. The statement required to access this instance is `Tcl& tel = Tcl::instance();`

4.4.2 Invoking OTcl Procedures

There are four different methods to invoke an OTcl command through the instance, `tcl`. They differ essentially in their calling arguments. Each function passes a string to the interpreter that then evaluates the string in a global context. These methods will return to the caller if the interpreter returns `TCL_OK`. On the other hand, if the interpreter returns `TCL_ERROR`, the methods will call `tkerror{}`. The user can overload this procedure to selectively disregard certain types of errors.

1. **Passing Results to/from the Interpreter :** When the interpreter invokes a C++ method, it expects the result back in the private member variable, `tcl-> result`.
2. **Error Reporting and Exit:** This method provides a uniform way to report errors in the compiled code.

4.5 COMMAND METHODS: DEFINITION AND INVOCATION

For every `TclObject` that is created, `ns` establishes the instance procedure, `cmd{}`, as a hook to executing methods through the compiled shadow object. The procedure `cmd{}` invokes the method `command()` of the shadow object automatically, passing the arguments to `cmd{}` as an argument vector to the `command()` method. The user can invoke the `cmd {}` method in one of two ways, by explicitly invoking the procedure, specifying the desired operation as the first argument, or implicitly, as if there were an instance procedure of the same name as the desired operation. Most simulation scripts will use the latter form.

4.6 TRACE ANALYSIS

This section shows a trace analysis. Running the TCL script generates a NAM trace file that is going to be used as an input to NAM and a trace file called "out.tr" that will be used

for our simulation analysis. Figure 4.3 shows the trace format and example trace DATA from "out.tr". Where each line in trace file represents an event associated to a packet.

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id
-------	------	-----------	---------	----------	----------	-------	-----	----------	----------	---------	--------


```

r : receive (at to_node)
+ : enqueue (at queue)          src_addr : node.port (3.0)
- : dequeue (at queue)         dst_addr : node.port (0.0)
d : drop      (at queue)

r 1.3556 3 2 ack 40 ----- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ----- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ----- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ----- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ----- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ----- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ----- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ----- 2 1.0 3.1 157 207

```

Figure 4.3 Trace format example

Each trace line starts with an event (+, -, d, r) descriptor followed by the simulation time (in seconds) of that event, and from and to node, which identify the link on which the event occurred information in the line before flags (appeared as "-----" since no flag is set) is packet type and size (in Bytes). The next field is flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script. Even though fid field may not be used in a simulation, users can use this field for analysis purposes.

The next two fields are source and destination address in forms of "node.port". The next field shows the network layer protocol's packet sequence number. The last field shows the unique id of the packet.

4.7 NETWORK ANIMATOR (NAM)

Network Animator (NAM) is an animation tool for viewing network simulation traces. It supports topology layout, packet level animation and various DATA inspection tools.

Before starting to use NAM, trace file need to be created. This trace file is usually generated by NS. It contains topology information, e.g. nodes and links, as well as packet traces. During a simulation, the user can produce topology configurations, layout information and packet traces using tracing events in NS.

Once the trace file is generated, NAM can be used to animate it. Upon startup, NAM will read the trace file, create topology, pop up a window, do layout if necessary and then pause at time 0. Through its user interface, NAM provides control over many aspects of animation. In Figure 4.4 a screenshot of a NAM window is shown, where the most important function are explained. Although the NAM software contains bugs, as do the NS software, it works fine most of the times and times and causes only little trouble. NAM is an excellent first step to check .

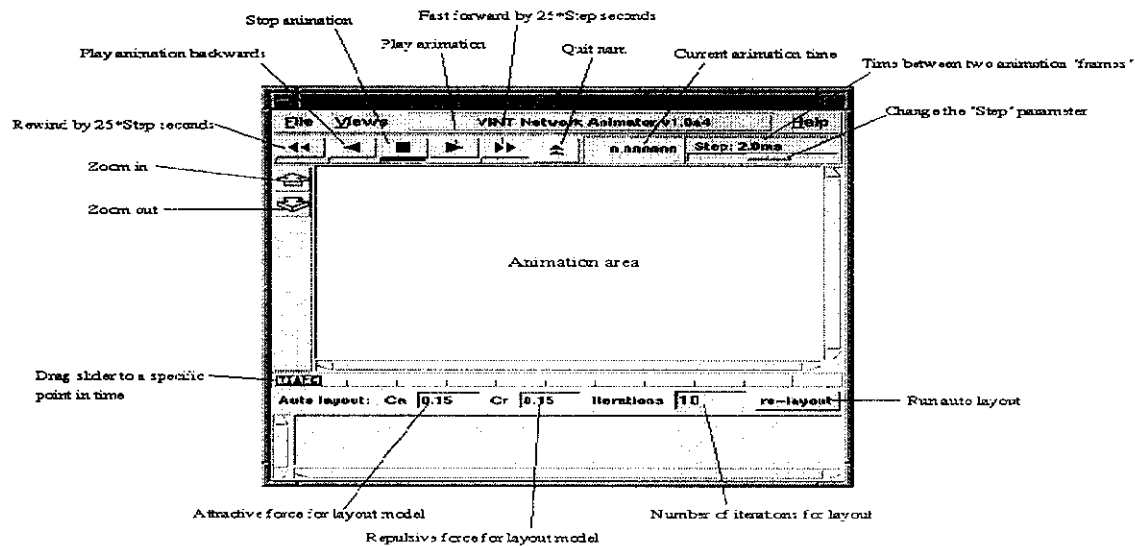


Figure 4.4 Screenshot of a NAM window

CHAPTER 5

PROJECT DESCRIPTION

5.1 PROBLEM DEFINITION

In wireless sensor network, power consumption is the one of the main obstacle to degrade the performance of wireless sensor nodes. After placed the sensor node in the environment the energy is losing drastically due to idle listening, packet drop, mobility. Since increase the lifetime of the sensor network use the duty cycling mechanism to overcome the power consumption. The proposed mechanism is used to solving the problem of power consumption by using the scheduling protocol. The proposed protocol is created by modifying the existing protocol AODV.

5.2 OVERVIEW OF THE PROJECT

In wireless sensor networks a data collection scenario where data typically flow from source nodes to the sink, while data from the sink to the sources are much less frequent. Assume that nodes are organized to form a logical routing tree (or data gathering tree) rooted at the sink, and use an underlying CSMA (Carrier Sense Multiple Access) MAC protocol for communication. These assumptions are quite realistic, as most MAC protocols commonly used in WSNs are CSMA-based, and many popular routing protocols for WSNs rely on a routing tree. In a real deployment the routing tree is re-computed periodically to cope with possible topology changes and better share the energy consumption among nodes. However, as nodes are supposed to be static, can assume that the routing tree remains stable for a reasonable amount of time. also assume that sensor nodes are synchronized through some synchronization protocol.

The communication between a parent and its children occurs in communication periods which repeat periodically. Each communication period includes an active interval during which nodes communicate by using the underlying MAC protocol, and a silence interval during which nodes turn their radio off to save energy. Active intervals are staggered so that nodes at the lower levels in the routing tree wake up earlier than their ancestors.

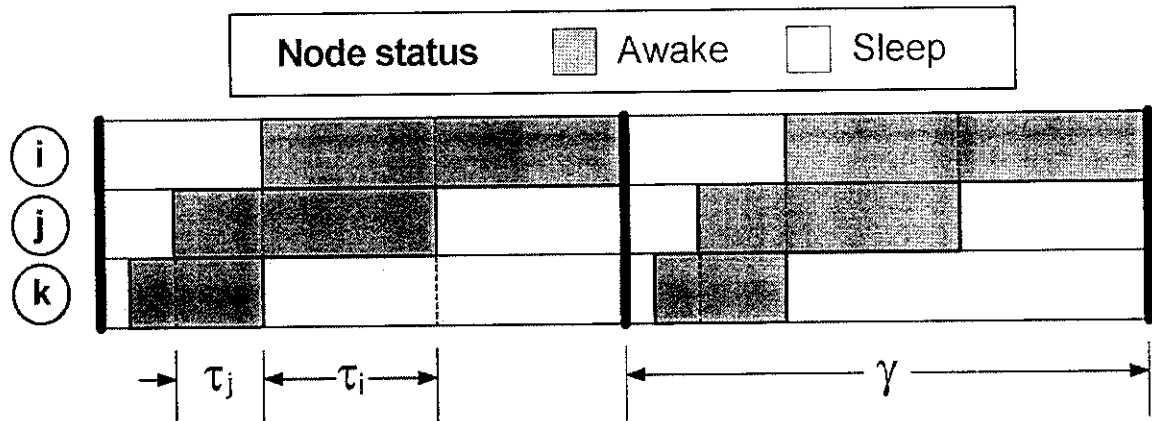


Figure 5.1 Parameters of the sleep scheduling protocol

The active interval of each (intermediate) sensor node consists of two adjacent talk intervals (TI), the first one with its children and the other one with its parent². Throughout, refer to the talk interval shared by a generic node j and all its children, during the m -th communication period. Although parent nodes can independently set their talk interval, a collective effort is needed for the schedule of the whole network to remain consistent and energy efficient. Hence, as a result of a change in the talk interval of a single parent node, the network-wide schedule must be rearranged. This is accomplished by appropriately shifting active intervals of a number of nodes, so as to ensure that (i) the active intervals of all nodes are properly staggered, and (ii) the two talk intervals of each node are contiguous. Two special messages, direct beacons and reverse beacons, are used for propagating schedule parameters to downstream and upstream nodes, respectively.

Direct beacons are broadcast by every parent node to all its children during each communication period. Instead, reverse beacons are sent in the opposite direction, i.e., from a child to its parent. As it will be shown below, direct beacons are critical for the correctness of the protocol. Hence, scheduling protocol also includes mechanisms for (i) increasing the probability of successful delivery of direct beacons, and (ii) enforcing a correct (even if non optimal) behavior of nodes in case they miss a direct beacon. In particular, to increase the probability of successful delivery, direct beacons are transmitted at the end of each talk interval, in a reserved time period (Beacon Period).

5.3 MODULE DESCRIPTION

5.3.1 TALK INTERVAL PREDICTION

In the scheduling protocol a sleep schedule is basically defined by the communication period and the talk interval of each individual parent node. The length of the communication period is closely related to the specific application and, thus, it is a global parameter specified by the sink when distributing the query. A variation in the communication period corresponds to a modification of the query, i.e. the new interval for the periodic data acquisition.

Choosing an appropriate talk interval is somewhat more involved. Ideally, each parent node should set the talk interval with its children to the minimum time needed to successfully receive all messages from all children. However, this time depends on a number of factors such as the underlying MAC protocol, channel conditions, degree of contention, number of messages to be received, and so on. In its turn, the number of messages to be received by a sensor node depends on the number of its children, the message generation rate at source nodes, and the network topology. Therefore, computing the ideal talk interval would require the global knowledge of the network. Moreover, this value should be continuously updated as the operating conditions change over time. Since such an approach is not practical, propose here an adaptive technique that approximates this ideal scheme by letting every parent node to choose autonomously its own talk interval with its children. The decision involves only local information and, thus, it does not require to know the network topology.

In principle, any algorithm can be used to estimate the expected talk interval in the next communication period. Here used the simple algorithm discussed below. Each parent node measures and stores the following quantities.

- Message inter-reception time (m). This is the difference between the time instants at which two consecutive messages are correctly received.
- Number of received messages (n_{pkt}). The total number of messages correctly received in a single communication period.

The time expected to get all messages sent by children in the next communication period is then estimated as $M \times n_{\text{pkt}}^{(\text{max})}$, where M and $n_{\text{pkt}}^{(\text{max})}$ are the average inter-reception time and the maximum number of received messages over the last L communication periods (observation window), respectively. Using $n_{\text{pkt}}^{(\text{max})}$ is a conservative choice to cope with possible message losses. The former time interval should be appropriately increased to allow the parent node to send a direct beacon at the end of talk interval. Finally, to reduce the number of possible values, the expected talk interval is divided into a number of time slots, whose duration is denoted by q . Hence, the expected talk interval for the $(m+1)$ -th communication period can be expressed as $\text{CP}(m+1) = \text{ceil}((n_{\text{pkt}}^{(\text{max})} + b)/q)q$, where b denotes the length of the Beacon Period, i.e., the time interval reserved for beacon transmission. The duration q of the time slot should be chosen as a trade-off between efficiency and stability. From one hand, a low value of q allows a fine granularity in setting the talk interval duration, but may introduce frequent changes in the sleep schedule. On the other hand, a large value of q makes the schedule more stable, but may lead to talk intervals larger than necessary, thus wasting energy. It may be worthwhile noting that the expected talk interval cannot be lower than one slot. This guarantees that any child has always a chance to send messages to its parent, even after a phase during which it had no traffic to send.

5.3.2 SLEEP COORDINATION

As anticipated, the sleep coordination algorithm is based on two special messages (direct and reverse beacons), and the sleep schedule rearrangement after a talk interval variation is accomplished by appropriately shifting the talk intervals of nodes. So, as to ensure that the network-wide schedule remains consistent. Direct beacons are broadcast at each communication period by every parent node during the Beacon Period, i.e., at the end of the talk interval with its children. They include the schedule parameters for the next communication period. Specifically, the direct beacon sent by a node in the m -th communication period contains:

- The length of the next communication period $\text{CP}^{(m+1)}$.
- Next wakeup time of parent node $t_{\text{parent},j}^{(m+1)}$
- The length of the next talk interval to be shared with its children $\text{TI}_j^{(m+1)}$

Conversely, *reverse beacons* are sent by child nodes. They may be sent at any time during the talk interval, and only include the amount of time the talk interval of the parent node has to be shifted. As schedules are local, nodes only have to coordinate with their parent, i.e., they have to know the wakeup time of their parent, and use it as a basis for establishing schedules with their children.

Assuming that:

1. clocks of nodes are properly synchronized and
2. direct and reverse beacons never get lost.

It can be shown that the following properties hold (the corresponding proofs are in:

Property 1 (Schedule agreement). Child nodes wake up at the instant, and for the duration, enforced by their parent, even when talk intervals change.

Property 2 (Nonoverlapping schedules). For any two nodes and such that is a child of , the talk intervals and are not overlapped.

Property 3 (Adjacent schedules). In steady-state conditions, the talk intervals shared by any node with its children and its parent, respectively, are contiguous. The above properties guarantee that, after a change has occurred in one or more talk intervals, the global sensor network is able to reach a new coordinated and energy-efficient schedule. In particular, Property 1 guarantees that activity times of a parent and its children are coordinated even after a schedule variation.

Property 2 ensures that talk intervals of different parent nodes remain properly staggered. Finally, Property 3 guarantees that, in steady state conditions, each sensor node wakes up and goes to sleep just once per communication period.

5.3.3 SCHEDULE ROBUSTNESS

5.3.3.1 BEACON PROTECTION

Beacon messages are critical for correctness of the protocol. When a node misses a direct beacon containing the new parameters, it cannot schedule its activity for the next communication period. In addition, the node cannot send direct beacons to its children until it reacquires the correct schedule information.

As a consequence, the loss of coordination propagates along the routing tree to its descendants. Direct beacons may get lost, for example, due to communication errors or collisions with other beacons or regular messages transmitted by interfering nodes. As direct beacons are sent through broadcast frames, they cannot be retransmitted by the underlying MAC protocol. Instead, reverse beacons are unicast messages and, thus, they are retransmitted by the MAC protocol if not received correctly. To add robustness to the direct beacon transmission and prevent collisions, the last part of the talk interval referred to as *Beacon Period* is reserved for the direct beacon transmission only. Child nodes must refrain from initiating regular message transmissions during the Beacon Period. In addition, the transmission of the direct beacon is initiated with a random backoff delay. Finally, two back-to-back copies of the direct beacon are transmitted.

Algorithm 1: Actions performed by a leaf node k during the m -th communication period(refer figure 5.1)

```

upon wakeup {
// talk interval with parent node  $j$ , starting at  $t_{\text{child},k}^{(m)}$ 
start timer ( $TI_j^{(m)}$ );
do {
    send data messages to parent node  $j$ ;
    wait for direct_beacon;
} until (timer expires or direct_beacon received);
if(direct_beacon ( $CP^{(m+1)}$ ,  $TI_j^{(m+1)}$ ,  $t_{\text{parent},j}^{(m+1)}$  ) received)
schedule next wakeup at  $t_{\text{child},k}^{(m+1)} = t_{\text{parent},j}^{(m+1)}$ ;
else {
     $CP^{(m+1)} = CP^{(m)}$ ;  $TI_j^{(m+1)} = TI_j^{(m)}$ ;
    schedule next wakeup at  $t_{\text{child},k}^{(m+1)} = t_{\text{child},k}^{(m)} + CP_{(m+1)}$ ;
}

```

```

}
start timer( $t_{\text{child},k}^{(m+1)} - t_{\text{now}}$ )
switch to sleep mode;
}

```

Algorithm 2: Actions performed by the sink node s during the m -th communication period(refer figure 5.1)

```

upon timer expiration {
// talk interval with children starting at  $t_{\text{parent},s}^{(m)}$ 
Shifts(m+1) = 0;
schedule direct_beacon transmission;
do {
    wait for messages from children;
    if(message == reverse_beacon(rshiftj(m+1)))
        shifts(m+1) = max(shifts(m+1), rshiftj(m+1))
    else if (message==data) update statistics;
} until (direct_beacon transmission time);
calculate TIs(m+1) from statistics;
schedule the beginning of next talk interval at:
 $t_{\text{parent},s}^{(m+1)} = t_{\text{parent},s}^{(m)} + CP^{(m+1)} + \text{shift}_s^{(m+1)} + \max(0, TI_s^{(m)} - TI_s^{(m+1)});$ 
send direct_beacon( $CP^{(m+1)}$ ,  $TI_s^{(m+1)}$ ,  $t_{\text{parent},s}^{(m+1)}$ )
start timer ( $t_{\text{parent},s}^{(m+1)} - t_{\text{now}}$ )
}

```


Algorithm 3: Actions performed at the m -th communication period by an intermediate node j having node i as its parent (refer figure 5.1).

```

Upon wakeup as a parent {
// talk interval with children starting at  $t_{\text{parent},j}^{(m)}$ 
Shift $_j^{(m+1)} = 0$ ;
Schedule direct_beacon transmission;
do {
wait for messages from a generic child node  $k$ ;
if(message == reverse_beacon(rshift $_k^{(m+1)}$ ))
shift $_j^{(m+1)} = \max(\text{shift}_j^{(m+1)}, \text{rshift}_k^{(m+1)})$ 
else if (message==data) {
update statistics;
store message in the local queue;
}
} until (direct_beacon transmission time);
calculate  $\text{TI}_j^{(m+1)}$  from statistics;
schedule the beginning of next talk interval at:
 $t_{\text{parent},j}^{(m+1)} = t_{\text{parent},j}^{(m)} + \text{CP}^{(m+1)} + \text{shift}_j^{(m+1)} + \max(0, \text{TI}_j^{(m)} - \text{TI}_j^{(m+1)})$ ;
send direct_beacon( $\text{CP}^{(m+1)}, \text{TI}_j^{(m+1)}, t_{\text{parent},j}^{(m+1)}$ );
}

```

```

Upon wakeup as a child {
// talk interval with parent node i starting at  $t_{\text{child},j}^{(m)}$ 

Start timer( $\text{TI}_i^{(m)}$ );
do {

if (  $\text{shift}_j^{(m+1)} > 0$  or  $\text{TI}_j^{(m+1)} - \text{TI}_j^{(m)}, 0$ )

// to node i;

    Send queued data messages to parent node i;

    Wait for direct_beacon from node i;

} until (activity_timer expires or direct_beacon received);

If(direct_beacon ( $\text{CP}^{(m+1)}, \text{TI}_i^{(m+1)}, t_{\text{parent},j}^{(m+1)}$ ) received)

Schedule next wakeup at  $t_{\text{child},j}^{(m+1)} = t_{\text{parent},j}^{(m+1)}$ ;

else {

 $\text{CP}^{(m+1)} = \text{CP}^{(m)}$ ;  $\text{TI}_i^{(m+1)} = \text{TI}_i^{(m)}$ ;

Schedule next wakeup at  $t_{\text{child},j}^{(m+1)} = t_{\text{child},j}^{(m)} + \text{CP}^{(m+1)}$ ;

}

Start timer( $t_{\text{child},j}^{(m+1)} - t_{\text{now}}$ );

Switch to sleep mode;

}

```

CHAPTER 6

EXPERIMENTAL RESULTS AND DISCUSSION

6.1 SIMULATION SCENARIO

The Scheduling Protocol is implemented in the network simulator. Network simulator has comprehensive radio model and has been widely used for simulation studies of WSNs. IEEE 802.11 radio with a bit rate of 2 megabits per second(Mbps) and a transmission range of 250m. The simulation duration is 100s, and the node density is 5 nodes/radio range. For WSN traffic, a source generates 512-byte packets at a constant rate of two packets/s. Sink node able to find the path based on the talk interval and send the direct beacon packets at end of the talk interval.

6.2 PERFORMANCE EVALUATION

The performances analyzed are,

- Packet Delivery Ratio (PDR) – Number of packets delivered to the destination by the number of packets expected to be received.
- Delay – Number of packets sent by Average time received by all receivers.
- Forwarding Cost – Number of packets transmitted by total number of packets received by all members.
- Overhead – Total number of bytes transmitted at MAC layer including ACK in case of unicast transmission.
- Independent of single point failure - Another novelty of this hybrid routing protocols is that it avoids single-point of failure with incorporation of alternate root node.
- Energy and bandwidth efficient - The protocol uses considerable reduction in packet processing therefore minimize the usage of energy and bandwidth.

6.3 PERFORMANCE COMPARISON

6.3.1 AODV Protocol

The Ad-hoc On-Demand Distance Vector (AODV) routing protocol is designed for use in ad-hoc mobile networks. AODV is a reactive protocol: the routes are created only when they are needed. It uses traditional routing tables, one entry per destination, and sequence numbers to determine whether routing information is up-to-date and to prevent routing loops. An important feature of AODV is the maintenance of time-based states in each node: a routing entry not recently used is expired. In case of a route is broken the neighbours can be notified. Route discovery is based on query and reply cycles, and route information is stored in all intermediate nodes along the route in the form of route table entries. The following control packets are used: routing request message (RREQ) is broadcasted by a node requiring a route to another node, routing reply message (RREP) is unicasted back to the source of RREQ, and route error message (RERR) is sent to notify other nodes of the loss of the link. HELLO messages are used for detecting and monitoring links to neighbours.

6.3.2 Scheduling Protocol

Scheduling protocol is used in the scenario where periodically send the data to sink node. This protocol choose the path based on the talk interval and also use the direct beacon packet for scheduling the sleep/wakeup time of each sensor node. So scheduling protocol performance completely varied from the AODV. Here, sink does not use the hello packet for every communication. The direct packet send at the end of the previous communication. Through this scheduling information the life time of the sensor network has extended. Moreover, reducing the idle listening of the sensor nodes in wireless sensor networks.

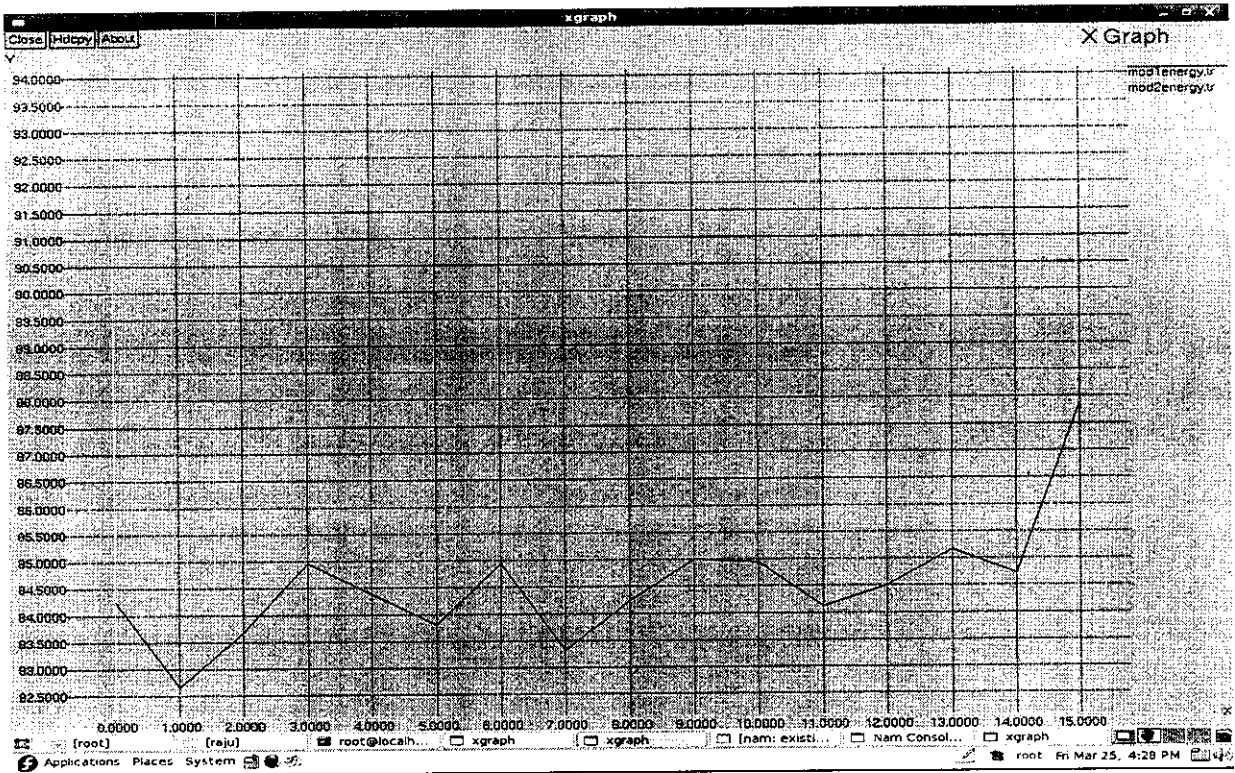


Figure 6.1 Comparison of AODV and Scheduling Protocol

CHAPTER 7

CONCLUSION AND FUTURE WORK

In wireless sensor network, sleep scheduling for efficient power management in wireless sensor networks targeted to periodic data acquisition. The proposed protocol has several strengths. It staggers the schedules of nodes according to their position in the routing tree.

This helps to reduce latency also when nodes are sleeping for most of the time and favors data aggregation. Unlike traditional staggered schemes, however, in the proposed protocol the active period of each sensor node is adjusted dynamically based on the traffic pattern and the operating conditions experienced by that node. Sleep schedule protocol is thus able to adapt to variations in the message generation rate, network topology, external conditions, and so on. In addition, as the active periods are tailored to the actual needs of each single node, the proposed protocol tends to minimize both energy consumption (thus extending the network lifetime) and message latency. Finally, the schedule protocol is conceived as an independent sleep/wakeup protocol operating above the MAC layer. Thus, it is independent from the underlying MAC protocol and can be used with different sensor platforms.

The system is more energy efficient in terms of number of path selection compared to other protocols. Still the future research can be made in the following directions:

- (a) More efficient schemes for reduce energy consumption to be considered supporting better resource tracking.
- (b) Intend to maintain a proper path management and Location updates.

APPENDIX I

SAMPLE SOURCE CODE

```

=====
# Existing system of Wireless Sensor Networks through AODV

# =====

#Routing table formation

# =====

# Define Node Configuration paramaters

=====

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)         50                          ;# max packet in ifq
set val(nn)             16                          ;# number of
mobilenodes
set val(rp)             AODV                        ;# routing protocol
set val(x)              1500                        ;# X axis distance
set val(y)              1500                        ;# Y axis distance
set opt(energymodel)    EnergyModel                ;# Initial Energy
set opt(radiomodel)     RadioModel                 ;# Transmission Model
set opt(initialenergy)  100                        ;# Initial energy in
Joules

# Creating Simulator Object
set ns [new Simulator]

# Creating NAM File
set namTracefile [open existing.nam w]
$ns namtrace-all-wireless $namTracefile $val(x) $val(y)

# Creating Multiple Trace
set traceFile [open mod2a.tr w]
$ns trace-all $traceFile

```

```

# Creating Topology
set topo [new Topography]
$stopo load_flatgrid $val(x) $val(y)

# Creating GOD(General Operation Director) Object
create-god $val(nn)

# Parameters
Phy/WirelessPhy set bandwidth_ 2e6
Phy/WirelessPhy set Pt_ 0.2818
Phy/WirelessPhy set freq_ 914e+6
Mac/802_11      set dataRate_ 2.0e6

puts " Number of Nodes = 16 "
puts " Source extraction "

# Configuring the Nodes in the Topology.
$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) \
    -energyModel $opt(energymodel) \
    -initialEnergy $opt(initialenergy) \
    -topoInstance $stopo \
    -agentTrace ON \
    -routerTrace ON \
    -macTrace ON \
    -movementTrace ON \
    -idlePower 0.2 \
    -rxPower 0.5 \
    -txPower 1.0 \
    -sleepPower 0.1 \
    -transitionTime 0.003 \
    -channelType $val(chan) \
    -maxSenRange 20 \
    -maxTransRange 40 \

#Creating Group-1 Nodes
set node(0) [$ns node]
set node(1) [$ns node]
set node(2) [$ns node]
set node(3) [$ns node]
set node(4) [$ns node]

# Parameters to the Node
for { set i 0 } { $i<16 } { incr i } {
    $node($i) random-motion 0
    $node($i) set X_ 0.0

```



```

$node($i) set Y_ 0.0
$node($i) set Z_ 0.0
$node($i) color black
$ns initial_node_pos $node($i) 30
}
#Displaying the nodes

set i
puts " The number of Nodes is 16"

for {set i 0} {$i <=9} {incr i} {
puts node_$i
}

$ns at 35.0 "$node(9) color black"
# Creating the Sink Agent

set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
set sink3 [new Agent/LossMonitor]
set sink4 [new Agent/LossMonitor]
set sink5 [new Agent/LossMonitor]

# Attatching the Sink with Node

$ns attach-agent $node(0) $sink0
$ns attach-agent $node(1) $sink1
$ns attach-agent $node(2) $sink2
$ns attach-agent $node(3) $sink3
$ns attach-agent $node(4) $sink4
$ns attach-agent $node(5) $sink5

# Creating the Source Agent

#Creating the Application/Traffic for Data Transmission

proc attach-CBR-traffic { node sink size interval } {
    #Get an instance of the simulator
    set ns [Simulator instance]
    #Create a CBR sink14 agent and attach it to the node
    set cbr [new Agent/CBR]
    $ns attach-agent $node $cbr
    $cbr set packetSize_ $size
    $cbr set interval_ $interval

    #Attach CBR source to sink;
    $ns connect $cbr $sink
    return $cbr
}

set cbr(0) [attach-CBR-traffic $node(0) $sink1 1024 .042]
set cbr(1) [attach-CBR-traffic $node(1) $sink4 1024 .042]

```

```

set cbr(2) [attach-CBR-traffic $node(1) $sink5 1024 .042]

set cbr(3) [attach-CBR-traffic $node(2) $sink5 1024 .042]
set cbr(4) [attach-CBR-traffic $node(2) $sink3 1024 .042]
set cbr(5) [attach-CBR-traffic $node(4) $sink7 1024 .042]
set cbr(6) [attach-CBR-traffic $node(4) $sink6 1024 .042]

set cbr(7) [attach-CBR-traffic $node(5) $sink7 1024 .042]
set cbr(8) [attach-CBR-traffic $node(5) $sink8 1024 .042]

set cbr(9) [attach-CBR-traffic $node(7) $sink10 1024 .042]
set cbr(10) [attach-CBR-traffic $node(7) $sink11 1024 .042]
set cbr(11) [attach-CBR-traffic $node(11) $sink14 1024 .042]

set cbr(12) [attach-CBR-traffic $node(8) $sink12 1024 .042]
set cbr(13) [attach-CBR-traffic $node(8) $sink9 1024 .042]

set cbr(14) [attach-CBR-traffic $node(12) $sink13 1024 .042]
set cbr(15) [attach-CBR-traffic $node(13) $sink15 1024 .042]
set cbr(16) [attach-CBR-traffic $node(14) $sink15 1024 .042]

```

```
# Finish Procedure to exec NAM Window
```

```

proc finish {} {

    global ns traceFile
    close $traceFile
    $ns flush-trace
        exec nam -r 0.2m existing.nam &
        exec xgraph mod2energy.tr &
        exec xgraph mod2energy.tr mod1energy.tr &
    exit 0
}

$ns at 35.0 "finish"
puts "Start of simulation.."
$ns run

```

```
// Find energy consumption
```

```

BEGIN {
node=0;
time=0.0;
energy=0.0;
}
{
if(FILENAME == "mod2a.tr") {
    node=$5;
    time=$3;
    energy=$7;
if((node==0)&&(time >= 34.997157 )) {
print node " " energy > "mod1energy.tr"
}
}
}

```

```

if((node==1)&&(time >= 34.997157 )) {
print node " " energy > "mod1energy.tr"
}
if((node==2)&&(time >= 34.997157 )) {
print node " " energy > "mod1energy.tr"
}
if((node==3)&&(time >= 34.997157 )) {
print node " " energy > "mod1energy.tr"
}
if((node==4)&&(time >= 34.997157 )) {
print node " " energy > "mod1energy.tr"
}
if((node==5)&&(time >= 34.997157 )) {
print node " " energy > "mod1energy.tr"
}

```

ene.awk file

```

BEGIN {
i=1
esum=0
avg=0
tim=0
}
{
if(FILENAME == "mod2a.tr") {
if($1=="N") {
if($3~tim) {
esum=esum+$7
i++
} else {

avg=esum/i
esum=0
i=1
if(avg!=0) {
print tim" "avg > "energy.xg"
}
tim++
}
}
}
}
END {
}

```

```

#=====
# proposed system of Wireless Sensor Networks through Scheduling protocol
# =====
#Routing table formation
# =====
# Define Node Configuration paramaters
#=====

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)        50                       ;# max packet in ifq
set val(nn)            16                       ;# number of
mobilenodes
set val(rp)            AODV                      ;# routing protocol
set val(x)              1500                    ;# X axis distance
set val(y)              1500                    ;# Y axis distance
set opt(energymodel)   EnergyModel              ;# Initial Energy
set opt(radiomodel)    RadioModel               ;# Transmission Model
set opt(initialenergy) 100                      ;# Initial energy in
Joules

# Creating Simulator Object
set ns [new Simulator]

# Creating NAM File
set namTracefile [open existing.nam w]
$ns namtrace-all-wireless $namTracefile $val(x) $val(y)

# Creating Multiple Trace
set traceFile [open mod2b.tr w]
$ns trace-all $traceFile

# Creating Topology
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Creating GOD(General Operation Director) Object
create-god $val(nn)

```

```

# Parameters
Phy/WirelessPhy set bandwidth_ 2e6
Phy/WirelessPhy set Pt_ 0.2818
Phy/WirelessPhy set freq_ 914e+6
Mac/802_11      set dataRate_ 2.0e6

puts " Number of Nodes = 15 "
puts " Source extraction "

# Configuring the Nodes in the Topology.
$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) \
    -energyModel $opt(energymodel) \
    -initialEnergy $opt(initialenergy) \
    -topoInstance $topo \
    -agentTrace ON \
    -routerTrace ON \
    -macTrace ON \
    -movementTrace ON \
    -idlePower 0.2 \
    -rxPower 2.5 \
    -txPower 2.0 \
    -sleepPower 0.1 \
    -transitionTime 0.003 \
    -channelType $val(chan) \
    -maxSenRange 20 \
    -maxTransRange 40 \

#Creating Group-1 Nodes
set node(0) [$ns node]
set node(1) [$ns node]
set node(2) [$ns node]
set node(3) [$ns node]
set node(4) [$ns node]
set node(5) [$ns node]

# Parameters to the Node
for { set i 0 } { $i<16 } { incr i } {
$node($i) random-motion 0
$node($i) set X_ 0.0
$node($i) set Y_ 0.0
$node($i) set Z_ 0.0
$node($i) color black
$ns initial_node_pos $node($i) 30
}
#Displaying the nodes

```

```

set i
puts " The number of Nodes is 10"

for {set i 0} {$i <=9} {incr i} {
puts node_-$i
}

# Creating the Sink Agent

set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
set sink3 [new Agent/LossMonitor]
set sink4 [new Agent/LossMonitor]
set sink5 [new Agent/LossMonitor]

# Attatching the Sink with Node

$ns attach-agent $node(0) $sink0
$ns attach-agent $node(1) $sink1
$ns attach-agent $node(2) $sink2
$ns attach-agent $node(3) $sink3
$ns attach-agent $node(4) $sink4
$ns attach-agent $node(5) $sink5

# Creating the Source Agent

#Creating the Application/Traffic for Data Transmission

proc attach-CBR-traffic { node sink size interval } {
    #Get an instance of the simulator
    set ns [Simulator instance]
    #Create a CBR sink14 agent and attach it to the node
    set cbr [new Agent/CBR]
    $ns attach-agent $node $cbr
    $cbr set packetSize_ $size
    $cbr set interval_ $interval

    #Attach CBR source to sink;
    $ns connect $cbr $sink
    return $cbr
}

set cbr(17) [attach-CBR-traffic $node(0) $sink1 1024 .042]
set cbr(18) [attach-CBR-traffic $node(1) $sink5 1024 .042]
set cbr(19) [attach-CBR-traffic $node(5) $sink7 1024 .042]
set cbr(20) [attach-CBR-traffic $node(7) $sink11 1024 .042]
set cbr(21) [attach-CBR-traffic $node(11) $sink14 1024 .042]

```

```

set cbr(22) [attach-CBR-traffic $node(14) $sink15 1024 .042]
set cbr(23) [attach-CBR-traffic $node(7) $sink12 1024 .042]
set cbr(24) [attach-CBR-traffic $node(12) $sink13 1024 .042]
set cbr(25) [attach-CBR-traffic $node(13) $sink15 1024 .042]

```

```
# signal transmission
```

```
# 0th node
```

```

$ns at 1.0 "$cbr(17) start"
$ns at 1.0 "$node(1) label Talkinterval"
$ns at 1.4 "$cbr(18) start"
$ns at 1.4 "$node(5) label Talkinterval"
$ns at 1.8 "$cbr(19) start"
$ns at 1.8 "$node(7) label Talkinterval"
$ns at 2.0 "$cbr(20) start"
$ns at 2.0 "$node(11) label Talkinterval"
$ns at 2.3 "$cbr(21) start"
$ns at 2.3 "$node(14) label Talkinterval"
$ns at 2.5 "$cbr(22) start"
$ns at 2.5 "$node(15) label Talkinterval"
$ns at 2.7 "$cbr(20) stop"
$ns at 2.7 "$cbr(21) stop"
$ns at 2.7 "$cbr(22) stop"

```

```
# Finish Procedure to exec NAM Window
```

```

proc finish () {
    global ns traceFile
    close $traceFile
    $ns flush-trace
    exec nam -r 0.2m existing.nam &
    exec xgraph mod2energy.tr &
    exec xgraph modlenergy.tr mod2energy.tr &
    exec ns graph.tcl &

    exit 0
}

```

```
$ns at 20.0 "finish"
```

```

puts "Start of simulation.."
$ns run

```

Bit error rate.awk

ber.awk

```

BEGIN {
drop=" ";

```

```

time=0.0;
m=0;

}
{
drop=$1;
time=$2;
if(drop=="D") {
m++;
printf("%f %d \n",time,m) > "bber.tr";
}
}
END {

}

```

Average delay.awk

```

BEGIN {
    for (i in send) {
        send[i] = 0
    }
    for (i in rcv) {
        rcv[i] = 0
    }
    delay = avg_delay = 0
    f=delaytrace
}

{
    read line
    if ($2 != "-t") {
        event = $1
        time = $2
        if (event == "+" || event == "-") node_id = $3
        if (event == "r" || event == "d") node_id = $4
        flow_id = $8
        pkt_id = $12
    }

    if ($2 == "-t") {
        event = $1
        time = $3
        node_id = $5
        flow_id = $39
        pkt_id = $41
    }

    if (event == "+" || event == "s") {
        send[pkt_id] = time
    }
}

```




```

        if ( event == "r") {
            recv[pkt_id] = time
        }
    }
END {
    for (i in recv) {
        if (send[i] == 0) {
            printf("\nError %g\n",i)
        }
        delay += recv[i] - send[i]
        num ++
    }

    if (num < 100 ) num +=1000
    if (num != 0) {
        avg_delay = delay / num
    } else {
        avg_delay = 0
    }
    printf( "AVG Delay=%10g \n",avg_delay)
}

```

Delivery ratio.awk

```

BEGIN {
    sendLine = 0;
    recvLine = 0;
}

{
    if ( $1 == "s" && ($7 == "cbr" || $7 == "tcp") && $4 == "AGT" )
    {
        sendLine++;
    }

    if ( $1 == "r" && ($7 == "cbr" || $7 == "tcp") && $4 == "AGT" )
    {
        recvLine++;
    }
    time = $2;

    if (( $1 == "s" && ($7 == "cbr" || $7 == "tcp") && $4 == "AGT" ) || ( $1
    == "r" && ($7 == "cbr" || $7 == "tcp") && $4 == "AGT" )) {
        printf($2 " " (recvLine/sendLine) "\n")
    }
}
END {
}

```

Throughput.awk

```
BEGIN {
    throughput = 0;
    sum = 0;
}

{
time = $2;
if ( $1 == "s" && ($7 == "cbr" || $7 == "tcp") && $4 == "AGT" )
{
    sendLine++;
}

if ( $1 == "r" && ($7 == "cbr" || $7 == "tcp") && $4 == "AGT" )
{
    recvLine++;
}

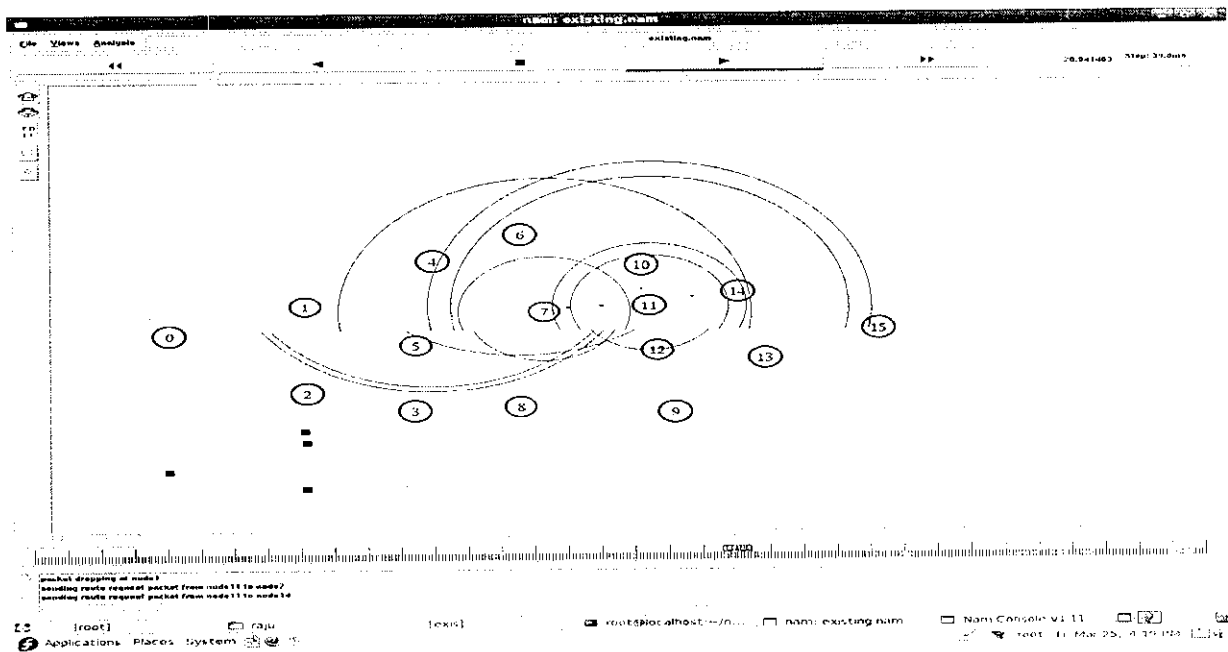
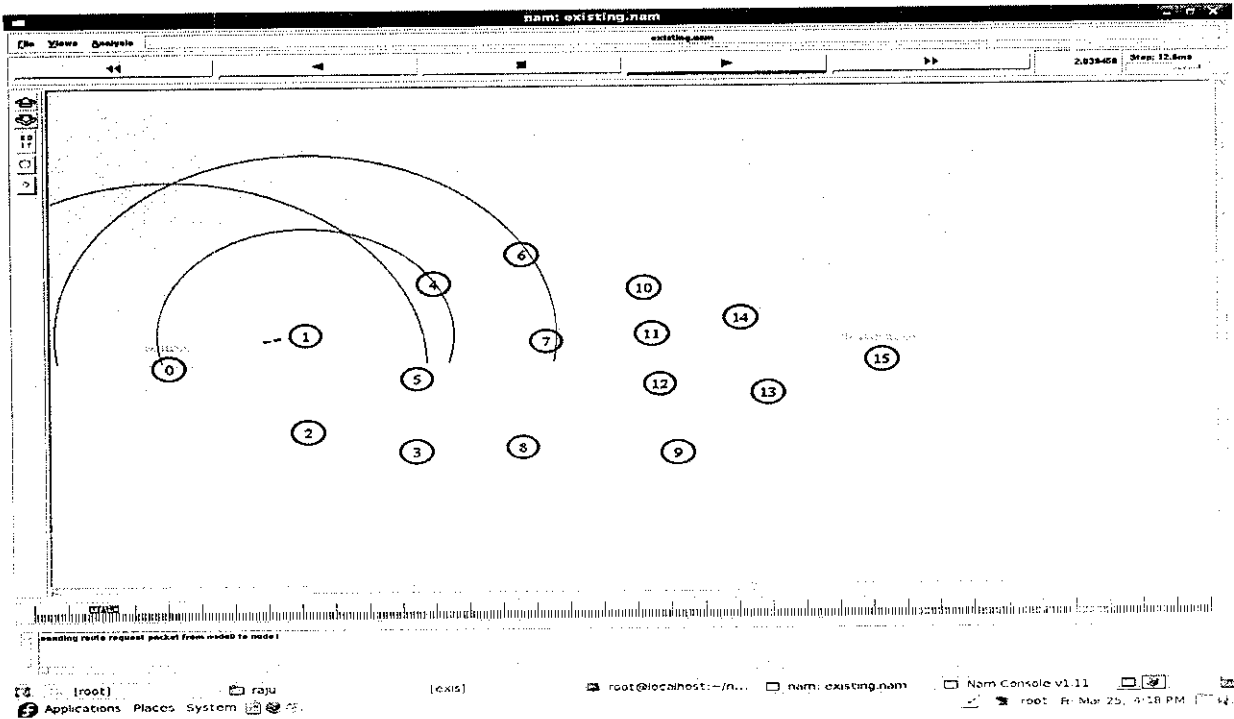
sum = sendLine+recvLine;
if (( $1 == "s" && ($7 == "cbr" || $7 == "tcp") && $4 == "AGT" ) || ( $1
== "r" && ($7 == "cbr" || $7 == "tcp") && $4 == "AGT" ))
{
    printf("%4f %4d \n",time,(sum/time)) > "throughputgraph.tr"
}

}
END {
}
```

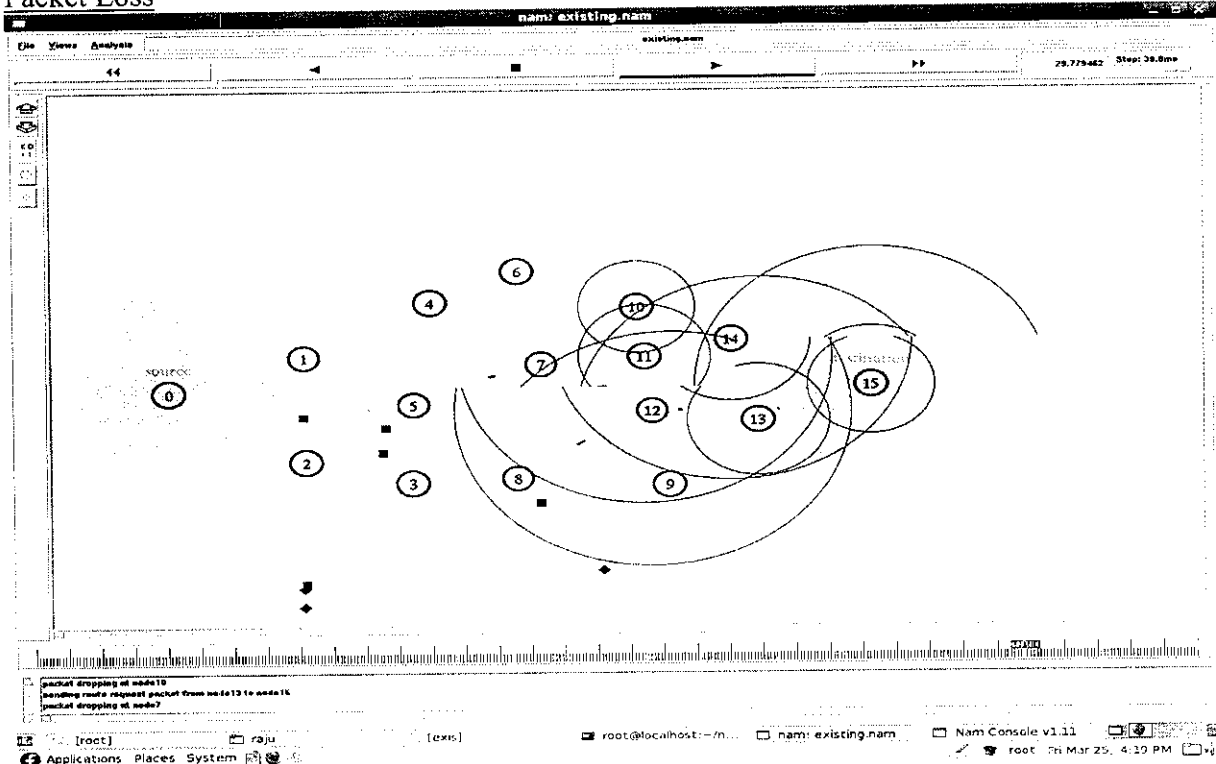
APPENDIX II

SNAP SHOTS

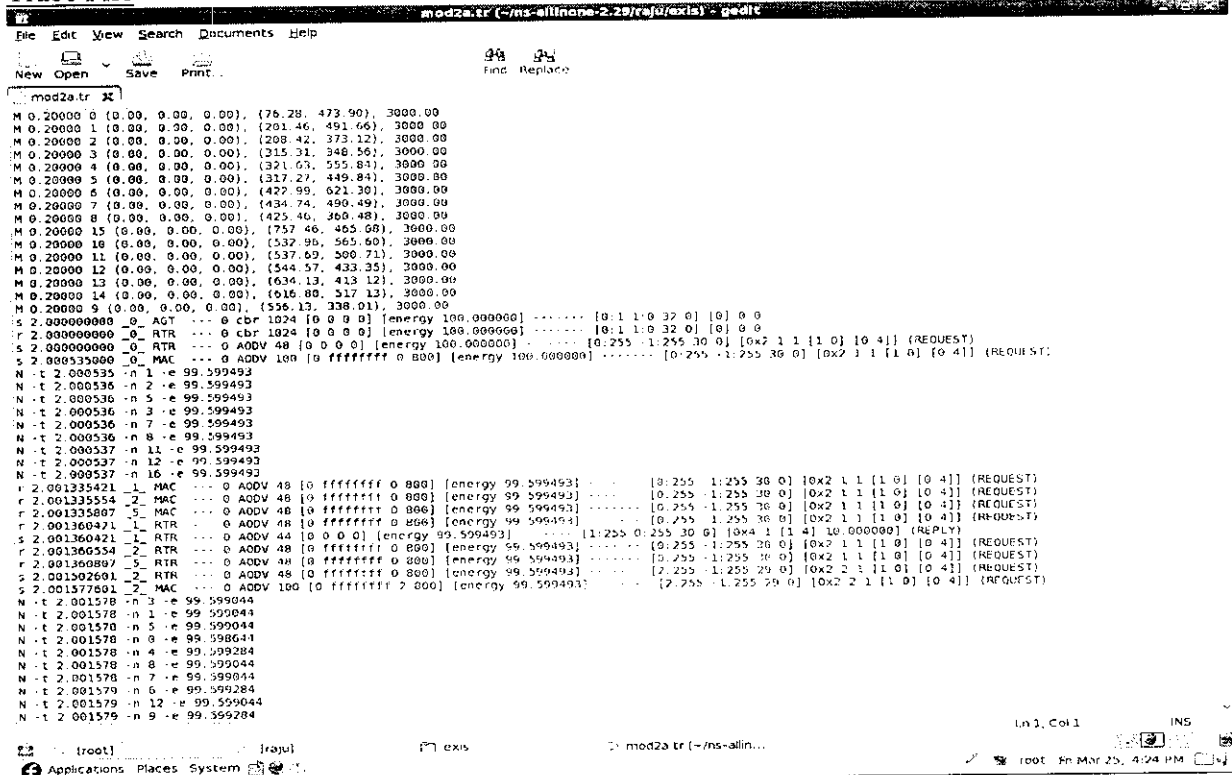
Existing System



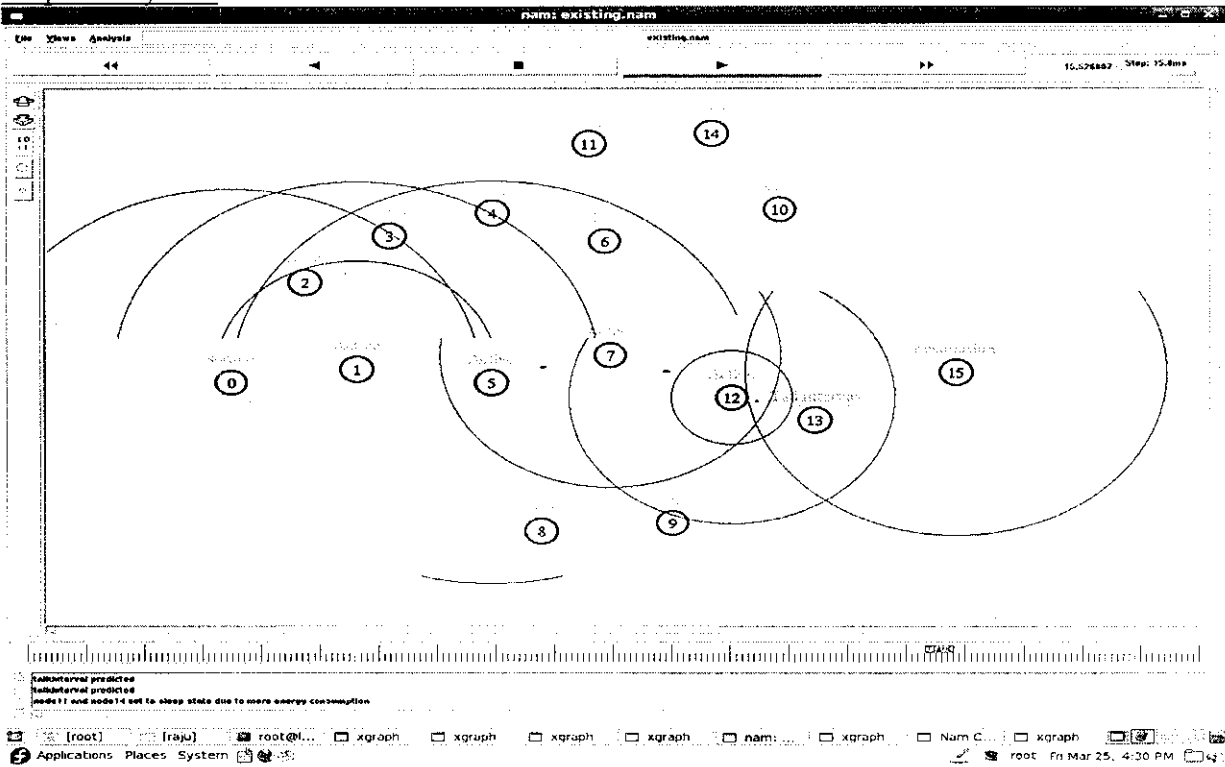
Packet Loss



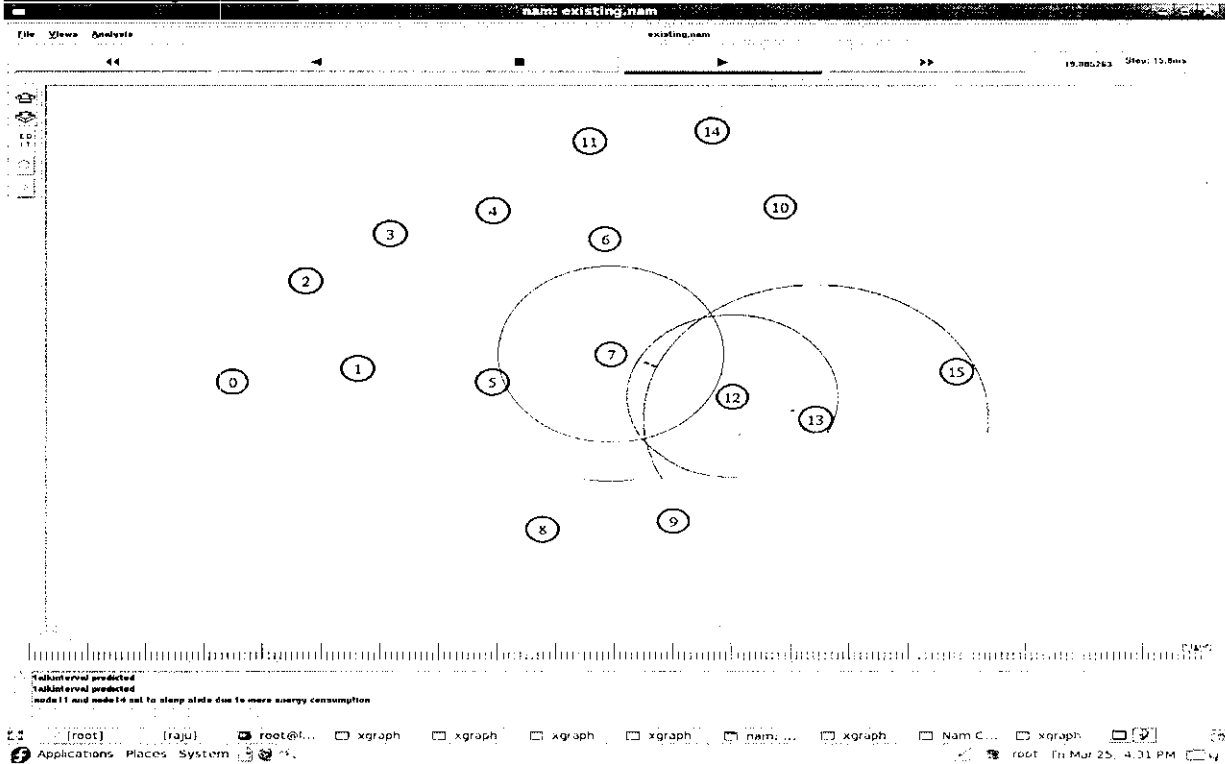
Trace File



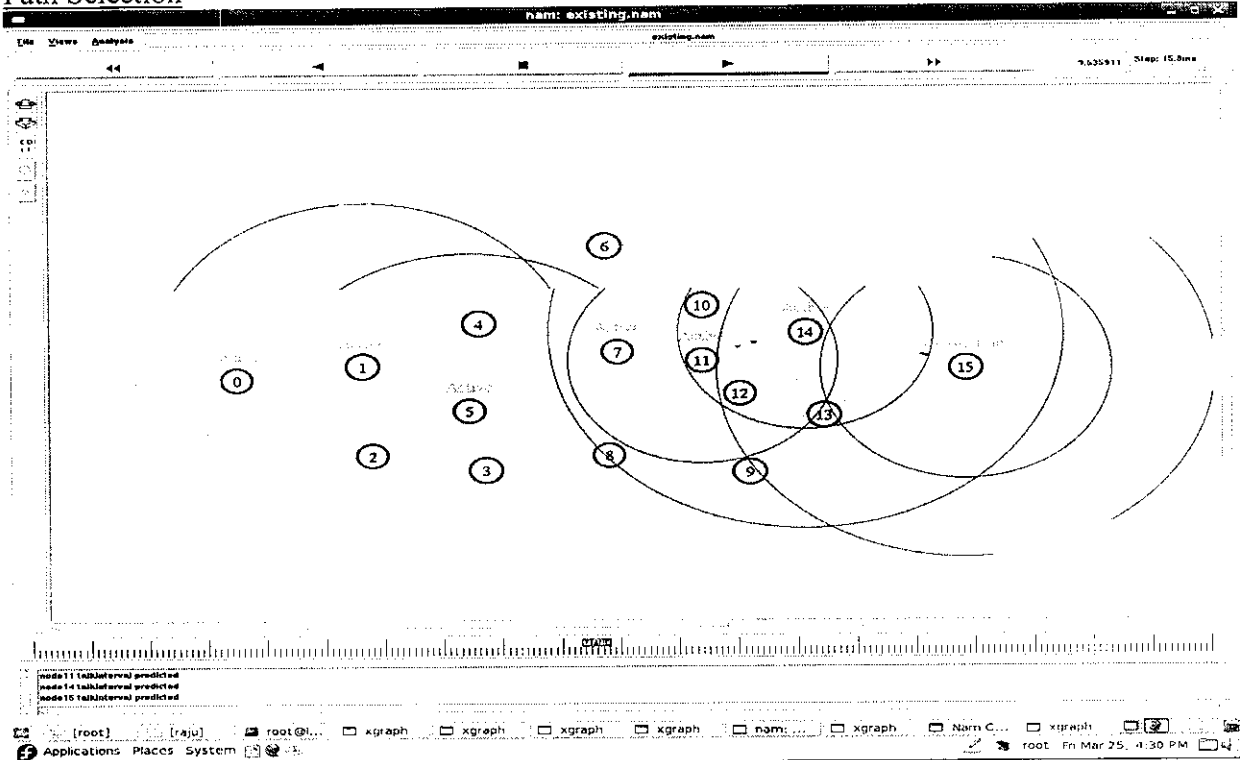
Proposed System



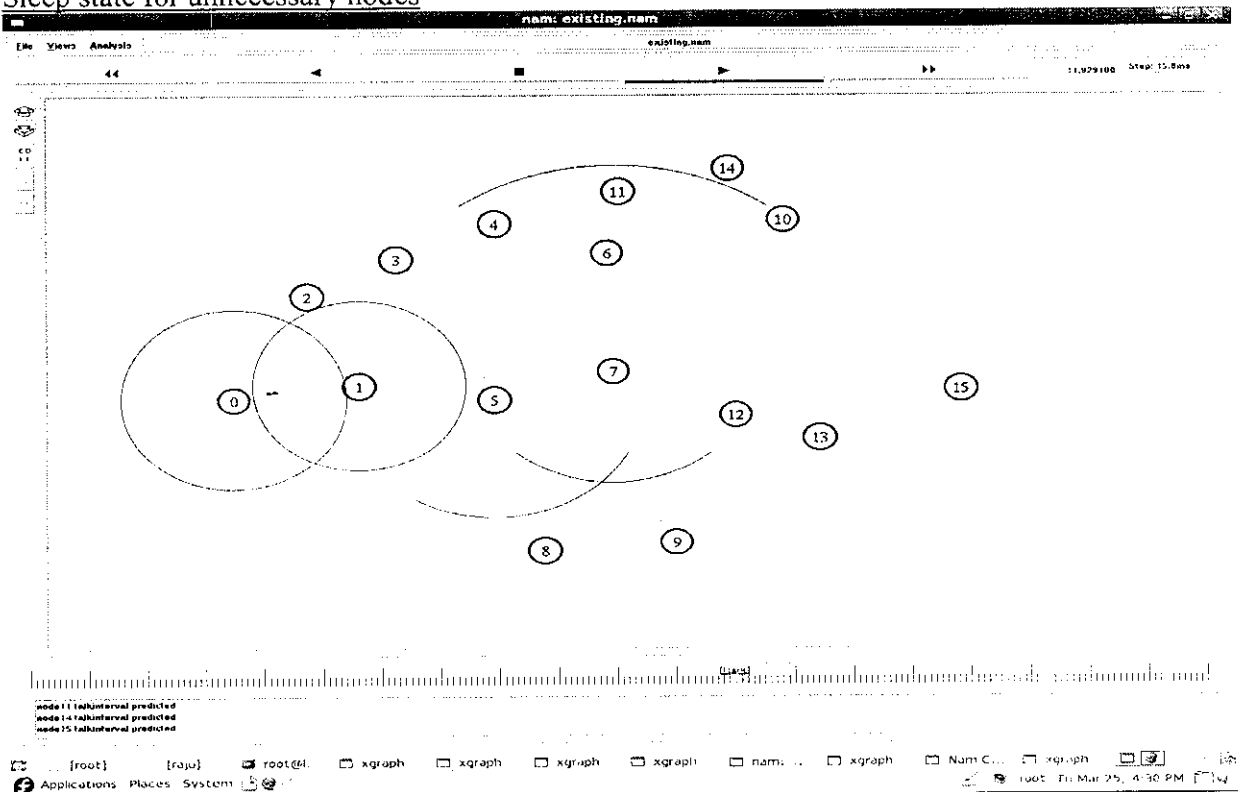
Talk interval prediction



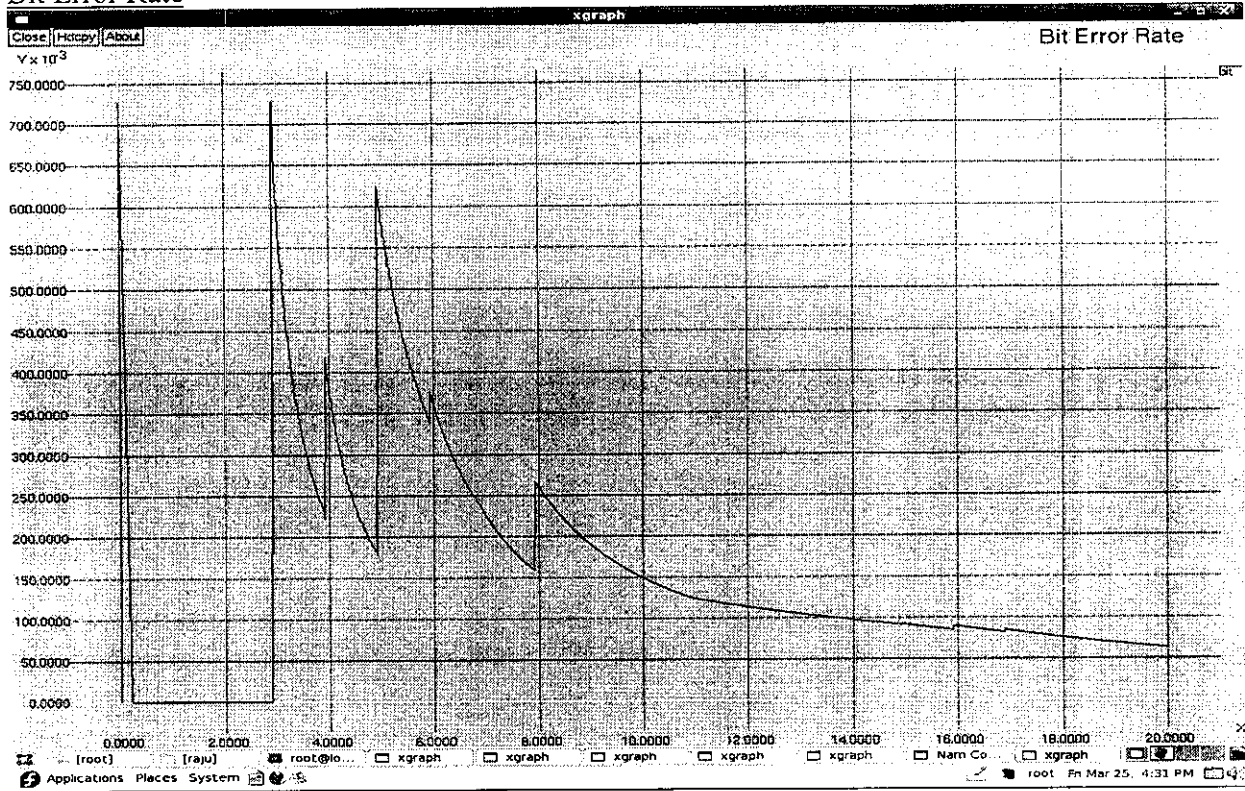
Path Selection



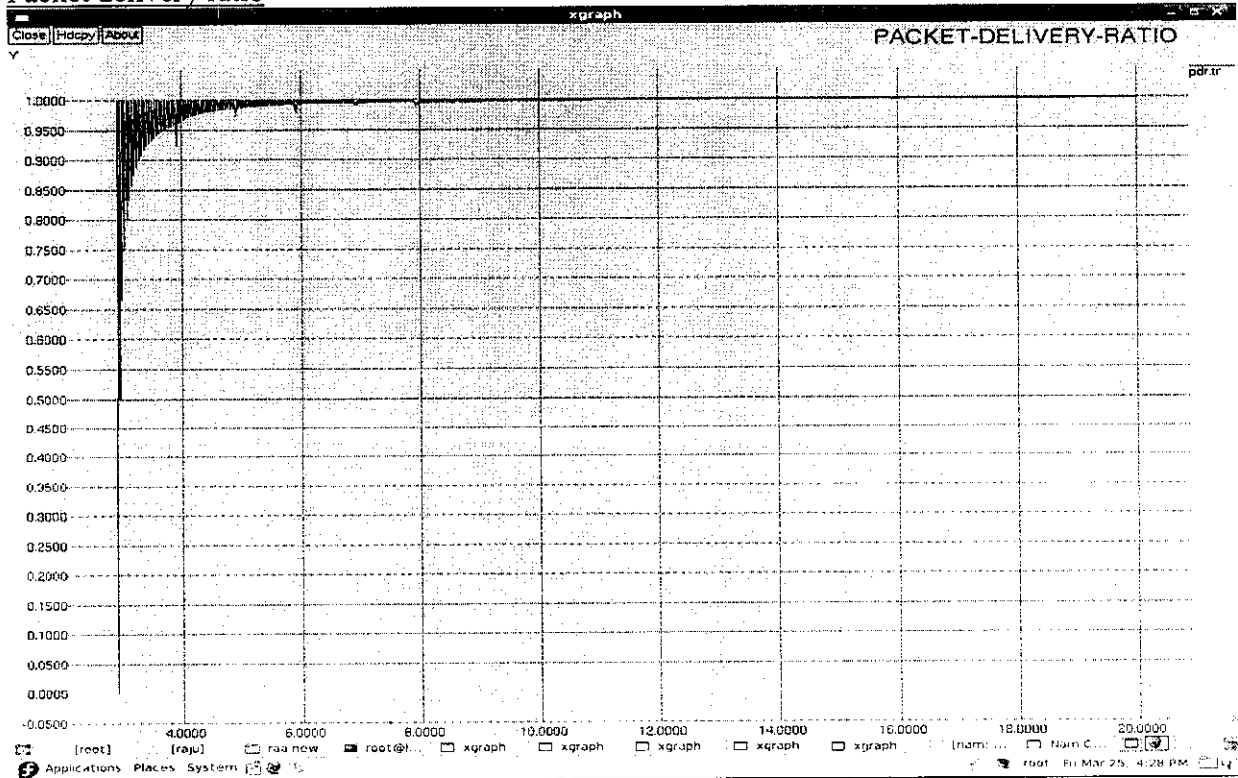
Sleep state for unnecessary nodes



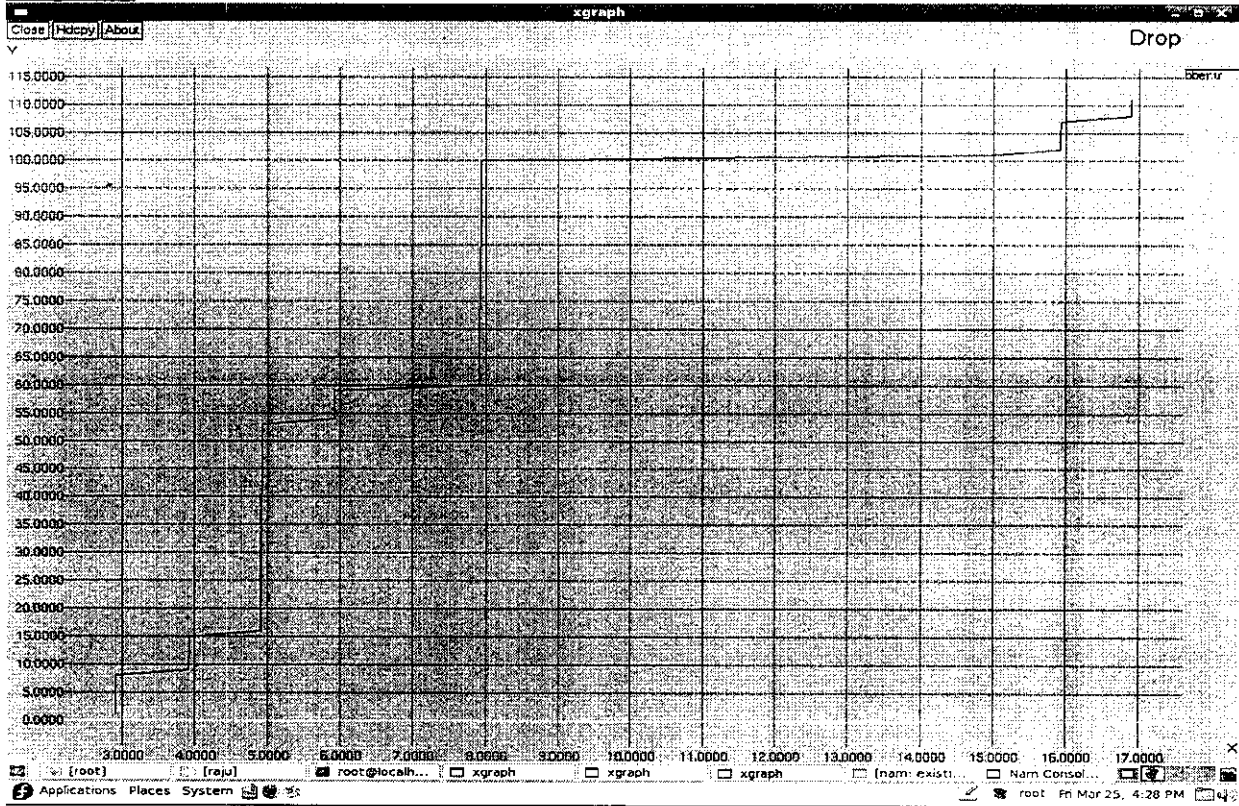
Bit Error Rate



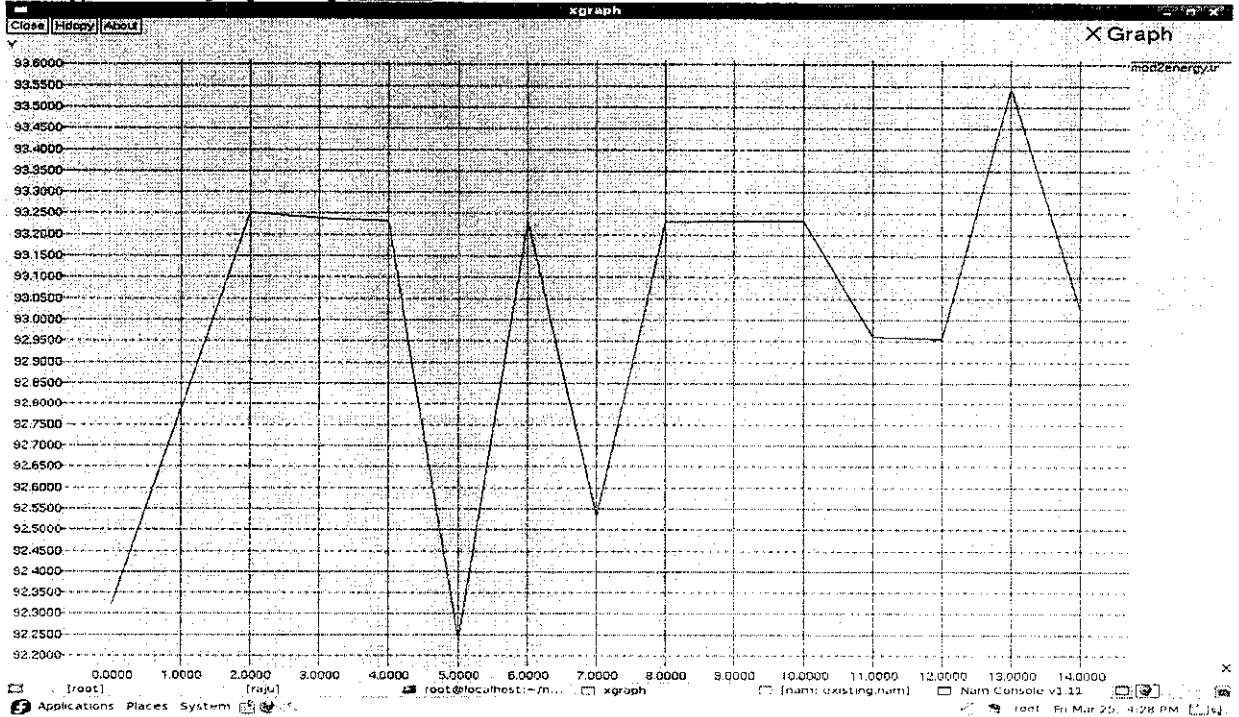
Packet delivery ratio



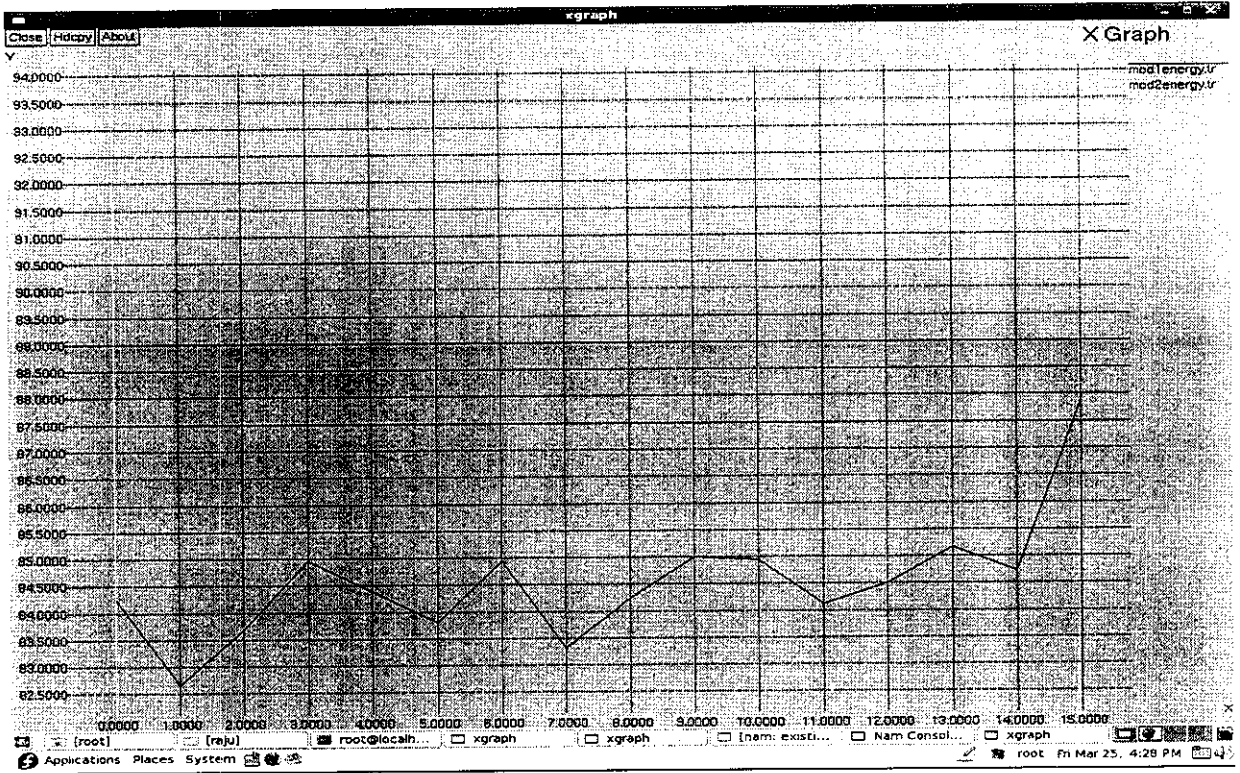
Drop Rate



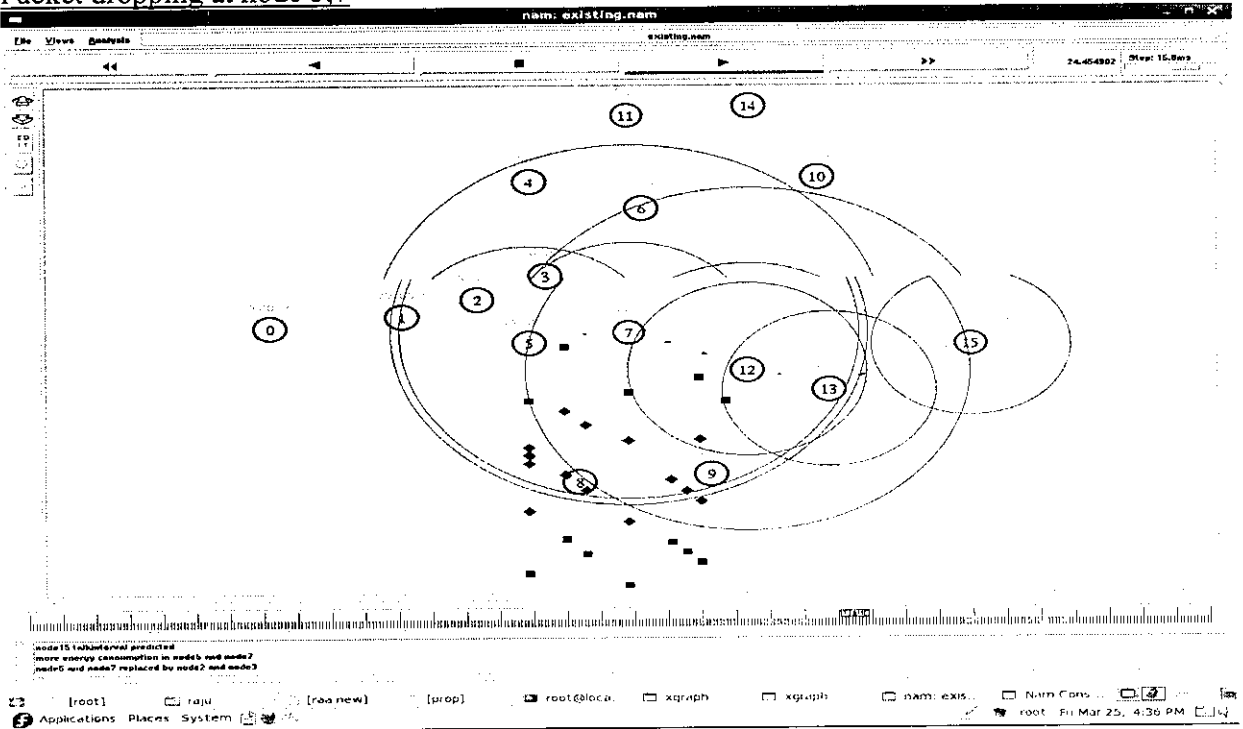
Energy level of proposed protocol



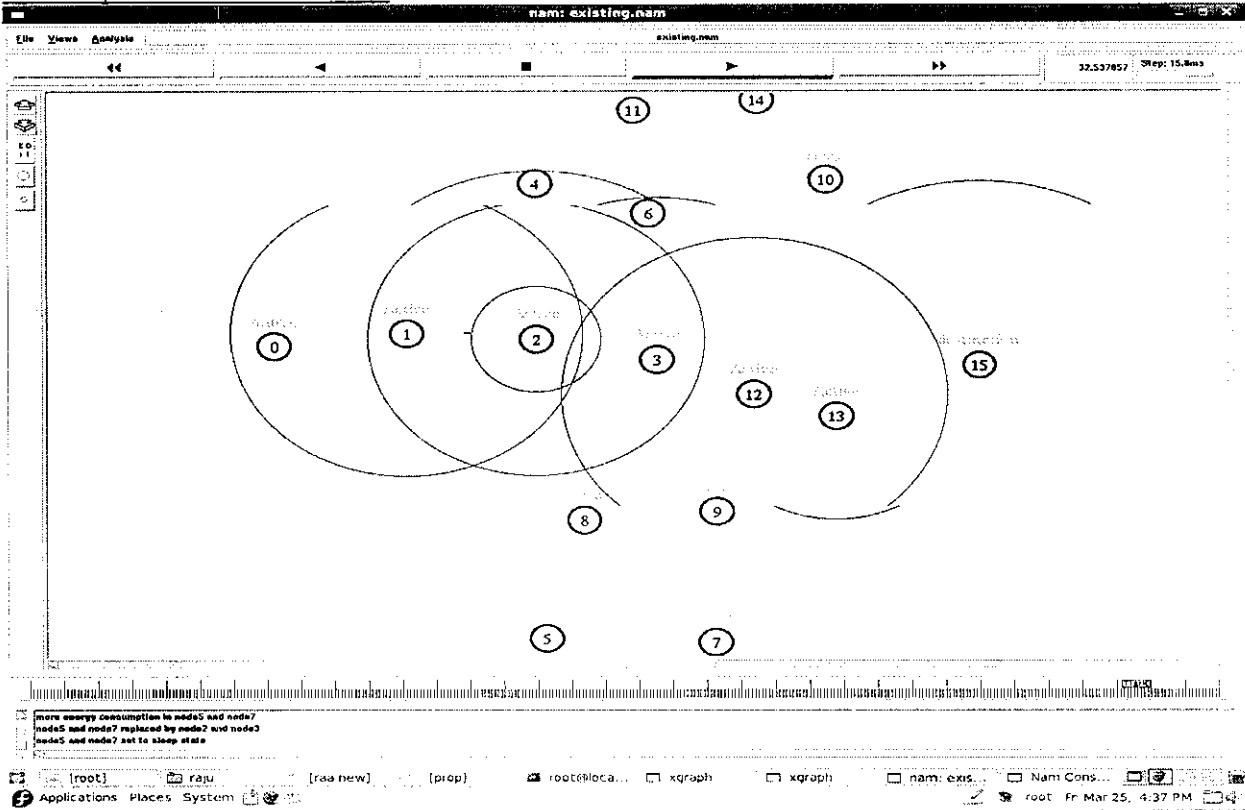
Comparison between AODV and Scheduling protocol



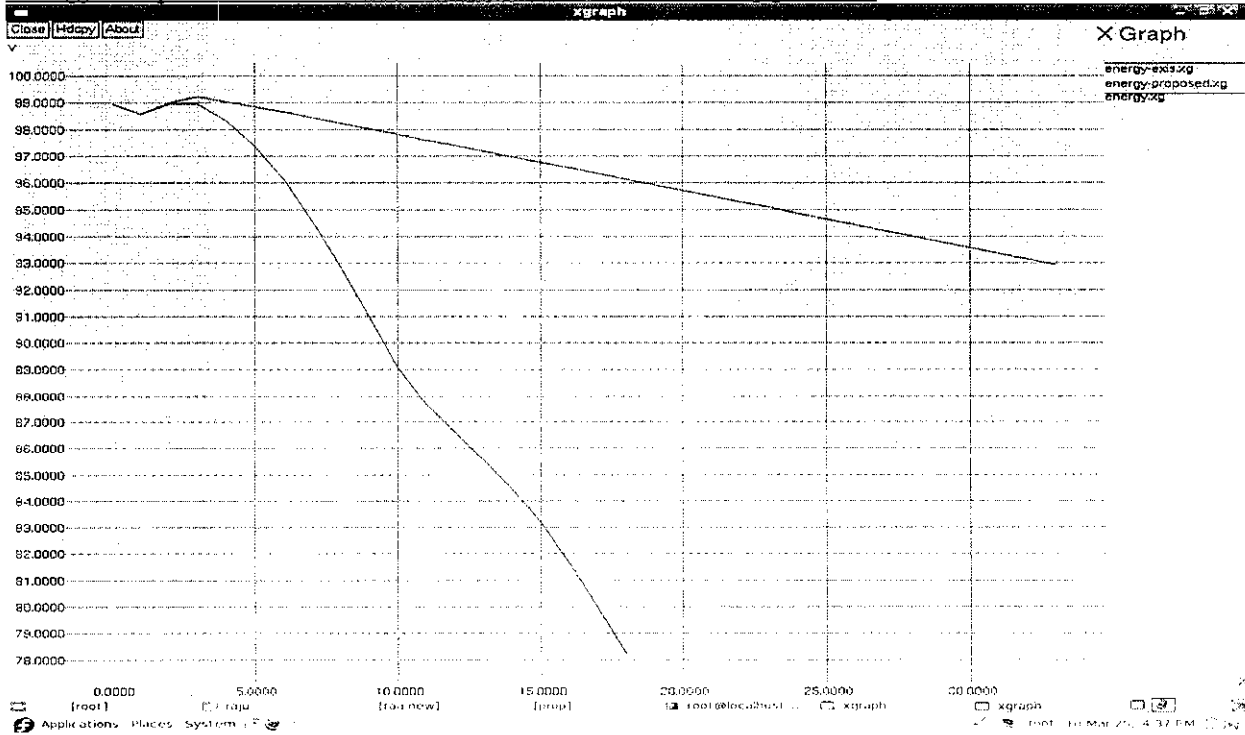
Packet dropping at node 5,7



Node replacement for node 5,7



Energy comparisons of AODV, Shifted Mehood, Scheduling protocol



REFERENCES

- [1] I.F. Akyildiz, Y. Sankarasubramaniam and E. Cayirci (2002), 'Wireless Sensor networks: a Survey', *Computer Networks*, Vol.38, N: 4.
- [2] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella (2009), 'Energy Conservation in Wireless Sensor Networks: a Survey', *Ad Hoc Networks*, Vol. 7, N.3.
- [3] K. Arisha, M. Youssef, M. Younis (2002) 'Energy-aware TDMA-based MAC for Sensor Networks', *Proc. IEEE IMPACCT '02*, New York City (USA).
- [4] D. Ganesan, A. Cerpa, W. Ye, Y. Yu, J. Zhao, D. Estrin (2004), 'Networking Issues in Wireless Sensor Networks', *Journal of Parallel and Distributed Computing*, Vol. 64 , pp. 799-814.
- [5] W. Heinzelman, A. Chandrakasan and H. Balakrishnan (2000), 'Energy-efficient Communication Protocol for Wireless Microsensor Networks', *Proc. HICSS-34*.
- [6] J. Heidemann, and D. Estrin (2002), 'An Energy-efficient MAC Protocol for Wireless Sensor Networks', *Proc. IEEE INFOCOM*.
- [7] H. Karl and A. Willig (2005), 'Protocols and Architectures for Wireless Sensor Networks', Chapter 10 (Topology Control).
- [8] M.D.Lemmon, Q.Ling,and Y. Sun (2003), 'Overload management in sensor-actuator networks used for spatially distributed control systems', *Proc. Conf. Embedded Networked Sensor Systems (SenSys)*, pp. 162–170.
- [9] K. S. Low, W. N. N. Win (2005), 'Wireless Sensor networks for Industrial Environments', *Proc. Int. Conf. Computational Intelligence for Modeling, Control and Automation (CIMCA 2005)*.

- [10] D. Miorandi, E. Uhlemann and S. Vitturi (2007), 'Guest Editorial: Special Section on Wireless Technologies in Factory and Industrial Automation', IEEE Transactions on Industrial Informatics, Vol. 3.
- [11] G.Platt, M.Blyde, S. Curtin, J. Ward (2005), 'Distributed Wireless Sensor Networks and Industrial Control Systems - a New Partnership', Proc. IEEE workshop on Embedded Networked Sensors (EmNetS-II), 2005.
- [12] V. Raghunathan, C. Schurgers, S. Park and M. B. Srivastava (2002), 'Energy Aware Wireless Microsensor Networks', IEEE Signal Processing Magazine, Vol. 19, No: 2.
- [13] P. Santi (2005), 'Topology Control in Wireless Ad Hoc and Sensor Networks', ACM Computing Survey, Vol. 37, pp. 164-194.
- [14] B.Sinopoli, C.Sharp, L. Schenato and S. S. Sastry(2003), 'Distributed Control Applications within Sensor Networks', Proc. of the IEEE, vol. 91, no. 8, pp. 1235–1246.
- [15] T. Van Dam and K. Langendoen (2003), 'An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks", Proc. ACM SenSys..
- [16] Willig (2008), 'Recent and Emerging Topics in Wireless Industrial Communications: a Selection', IEEE Transactions on Industrial Informatics, Vol. 4 N. 2.