P-3608

POWER BALANCING APPROACH TO OPTIMIZE THE COVERAGE TIME IN CLUSTERED WIRELESS SENSOR NETWORKS

PROJECT REPORT

Submitted by

M. PRAVEEN	0710108037
K. SANTHOSH	0710108043

In partial fulfillment for the award of the degree

of

BACHELOR 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: COIMBATORE-641 049

BONAFIDE CERTIFICATE

Certified that this project report entitled "**POWER BALANCING APPROACH TO OPTIMIZE THE COVERAGE-TIME IN THE CLUSTERED WIRELESS SENSOR NETWORKS**" is the bonafide work of Praveen. M and Santhosh. K who carried out the research under my supervision. Certified also, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible]	[illegible] 19/04/11
SIGNATURE	**SIGNATURE**
Mrs.P.Devaki.,M.E.,(Ph.D)	**Mr.K.Sivan Arul Selvan,M.E.,(Ph.D)**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
	Assistant Professor(SRG)
Department of Computer	Department of Computer
Science and Engineering	Science and Engineering
Kumaraguru College of Technology	Kumaraguru College of Technology
Coimbatore-641049	Coimbatore-641049

The candidate with University Register Nos. 0710108037 and 0710108043 were examined by us in the project viva-voce examination held on 20/4

[illegible]

INTERNAL EXAMINER

[illegible]

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled "**POWER BALANCING APPROACH TO OPTIMIZE THE COVERAGE-TIME IN THE CLUSTERED WIRELESS SENSOR NETWORKS**" is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any Institutions, for fulfillment of the requirement of the course study.

The report is submitted in partial fulfillment of the requirement for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University, Coimbatore.

Place: Coimbatore

Date: 19.04.2011

M. Praveen

(M.PRAVEEN)

[illegible]

(K.SANTHOSH)

ACKNOWLEDGEMENT

We are intend to express our heartiest thanks to our chairman **Arutselvar Dr.N.Mahalingam ,B.sc., F.I.E** and the Co-Chairman **Dr.B.K.Krishnaraj Vanavarayar** for given us this opportunity to embark on this project.

We would like to make a special acknowledgement and thanks to our correspondent **M.Balasubramaniam, M.com., M.B.A**., and our Director **Dr.J.Shanmugam** for his support and encouragement throughout the project.

We extend our sincere thanks to our Principal, **Dr. S.Ramachandran Ph.D.,** Kumaraguru College of Technology, Coimbatore, for being a constant source of inspiration and providing us with the necessary facility to work on this project.

We are indent to express my heartiest thanks to **Mrs.P.Devaki M.E,(Ph.D)**, *Project coordinator* ,Head of the Department of Computer Science &Engineering, for her valuable guidance and useful suggestions during the course of this project.

We express deep gratitude and gratefulness to **our Guide Mr.K.SivanArulSelvan M.E.,(Ph.D).,** *Assistant Professor*(SRG) Department of Computer Science & Engineering, for his supervision, enduring patience, active involvement and guidance.

We would like to convey our honest thanks to **all Faculty members** of the Department for their enthusiasm and wealth of experience from which we have greatly benefited.

We also thank our **friends and family** who helped us to complete this project fruitfully.

TABLE OF CONTENTS

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.

Such networks may be logically represented as a set of clusters by grouping together nodes that are in close proximity with one another. Cluster heads form a virtual backbone and may be used to route packets for nodes in their cluster. These nodes shoulder greater responsibility and may deplete their energy faster, causing them to drop out of the network. Therefore, there is a need for load-balancing among cluster heads to allow all nodes the opportunity to serve as a cluster head. In this project, we have investigated the maximization of the coverage time for a clustered wireless sensor network by optimal balancing of power consumption among cluster heads(CHs).

Clustering significantly reduces the energy consumption of individual sensors, but it also increases the communication burden on CHs. In the deterministic setup of the nodes, we consider the location of the nodes to be static and the sensor nodes are connected to the Cluster Head and the cluster head connects to the sink. In the stochastic network, the location of the nodes are analysed as a cone model and the routing between the cluster heads is done. Mechanisms are provided to balance the load in the networks by optimal cluster and routing method in both stochastic and deterministic setup. Simulation results are used to show how our approach can balance the load and improve the lifetime of the system.

LIST OF FIGURES

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 wireless network consisting of a large number of small sensors with low-power transceivers can be an effective tool for gathering data in a variety of environments. The data collected by each sensor is communicated through the network to a single processing centre that uses all reported data to determine characteristics of the environment or detect an event. The communication or message passing process must be designed to conserve the limited energy resources of the sensors. Clustering sensors into groups, so that sensors communicate information only to cluster heads and then the cluster heads communicate the aggregated information to the processing centre, may save energy. Randomized clustering algorithm to organize the sensors in a wireless sensor network into clusters. We then extend this algorithm to generate a hierarchy of cluster heads and observe that the energy savings increase with the number of levels in the hierarchy.

Why need for wireless sensor network? A Wireless Sensor Network (WSN) consists of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants. Inaccessible deployment scenarios, sensors are necessarily powered by energy-constrained, often non rechargeable batteries. This makes energy consumption a critical factor in the design of a WSN and calls for energy-efficient communication protocols that maximize the lifetime of the network.

Within a cluster, sensors transmit data to their CH, which in turn forwards the data to the sink, either directly or via a multi hop path through other (intermediate) CHs. Such an architecture is adopted by recent standard speci?cations for sensor networks. It signi?cantly reduces the battery drainage of individual sensors, which only need to communicate with their respective CHs over relatively short distances. It also has other advantages in terms of simplifying network management, improving security, and achieving better scalability. On the other hand, the clustering paradigm increases the burden on CHs, forcing them to deplete their batteries much faster than non-CH nodes. The additional energy consumption is attributed to the need to aggregate intra cluster traffic into a single stream that is transmitted by the CH and to relay inter-cluster traffic of other CHs. Such relaying is sometimes desirable because of its power-consumption advantage over direct (CH-to-sink) communication. Given the high density of sensors in common deployment scenarios, the traffic volume coming from a CH can be orders of magnitude greater than the traffic volume of an individual sensor. Even though the CH may be equipped with a more durable battery than the individual sensors it serves, the large difference in power consumption between the two can lead to shorter life time for the CH. Once the CH dies, no communications can take place between the sensors in that cluster until a new CH is selected.

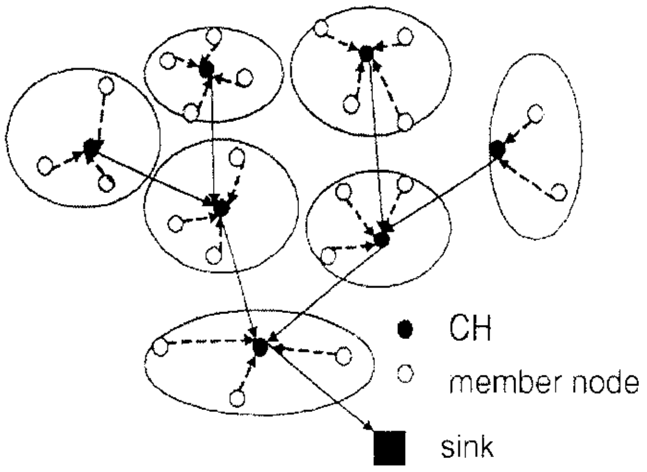


Fig. 1. Traffic implosion in WSNs.

For clusters with comparable area coverage and node density, the volume of intracluster traffic is roughly the same for all clusters. On the other hand, the traf?c relayed by different CHs is highly skewed; the closer a CH is to the sink, the more

traf?c it has to relay, and thus the faster it drains its energy reservoir. Such an imbalanced power consumption situation is essentially caused by the many-to-one communication paradigm in WSNs, i.e., traf?c from all sensors is eventually destined to the sink. If we donot take measures to deliberately balance power consumption at different CHs, a "traf?c implosion" situation will arise. More speci?cally, CHs that are closest to the sink will exhaust their batteries

?rst. Reassigning sensors to the next closest CHs to the sink will simply increase the energy consumption of these CHs. As a result, they will eventually be the second batch of CHs to run out of energy. This process continues to the next level of CHs, propagating from inside out and eventually leading to early loss of coverage and partitioning of the topology. So in this paper, we propose an efficient method of allocating the sensors of a cluster heads after it runs out of power.

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. If a power cable is used, many of the advantages enabled by the wireless communications are voided. 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.

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.

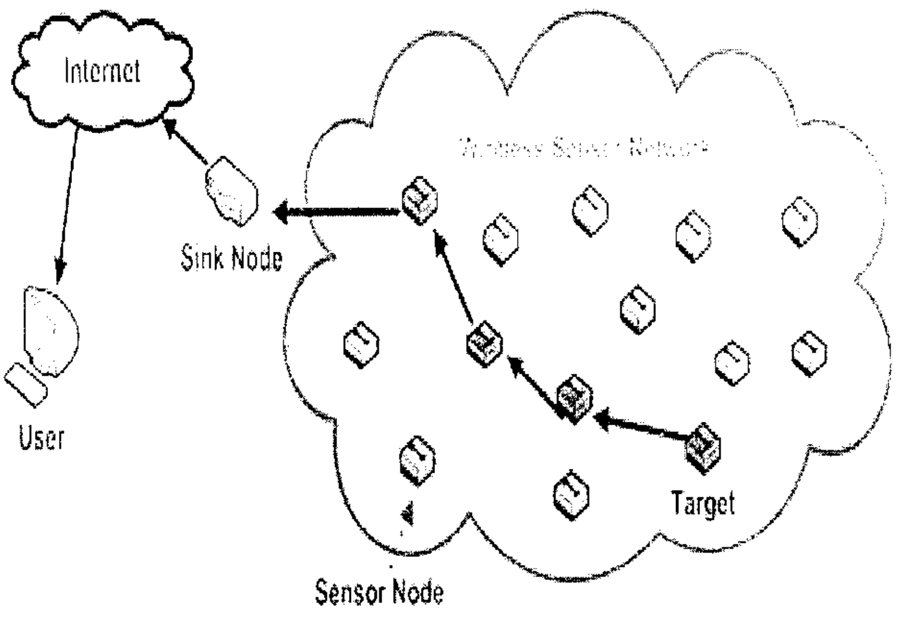


Figure 1.2 Function of Wireless Sensor Network

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.

The network is fault-tolerant because many nodes are sensing the same events. Further, the nodes cooperate and collaborate on their data, which leads to accurate sensing of events in the environment. The two most important operations in a sensor networks are data dissemination, that is, the propagation of data/queries throughput the network, and data gathering, that is, the collection of observed data from the individual sensor nodes to sink.

1.2 SENSOR NODE SYSTEM ARCHITECTURE

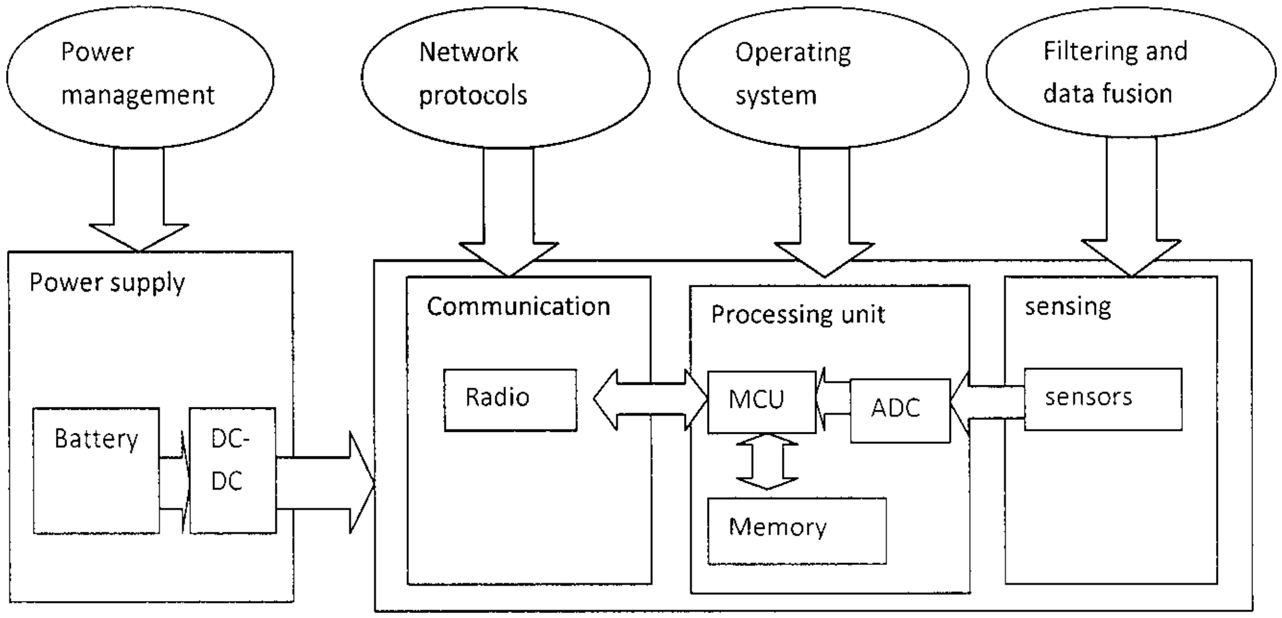


Figure 1.3 Sensor Node System Architecture

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.

The intrinsic properties of individual sensor nodes, pose additional challenges to the communication protocols in terms of energy consumption.

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, P - 3608

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, reach ability (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

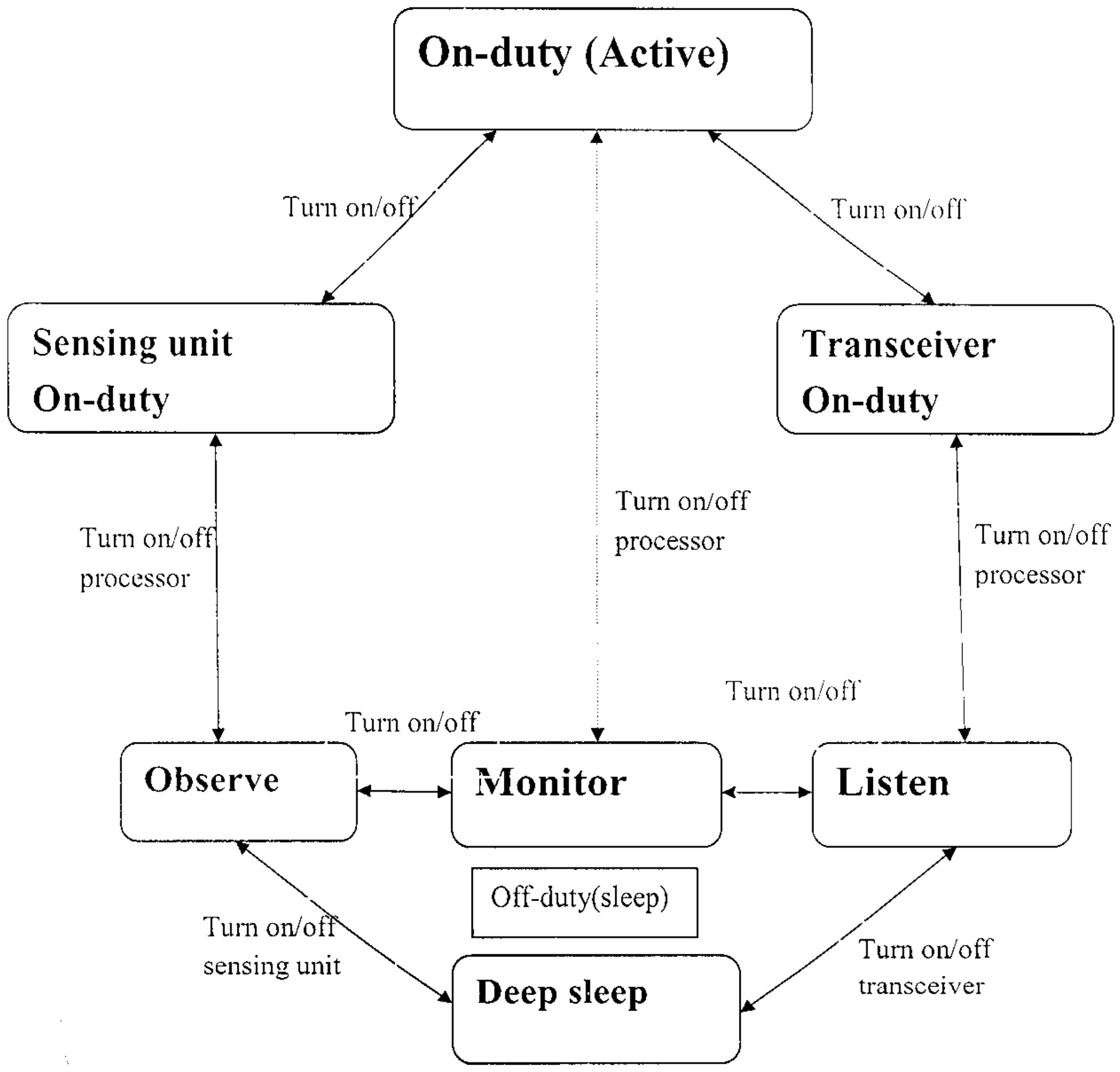


Figure 2.1 Power Saving Modes in Wireless Sensor Networks

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.

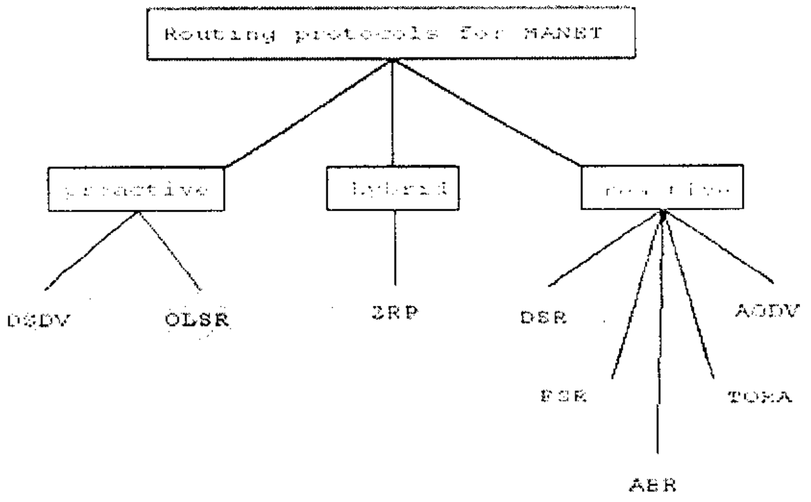


Figure 2.2 Routing Protocol

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 rediscovery 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 will 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 a, of the region of interest, is covered by at least one sensor. This is commonly called a-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.

Low Energy Adaptive Clustering Hierarchy

Early clustering algorithms mainly focused on the connectivity issue, aiming at generating the minimum number of clusters that ensures network connectivity. In these algorithms, the election of the CH is done based on node identity, connectivity degree, or connected dominating set.

Recently, there has been increased interest in studying energy-efficient clustering algorithms, in sensor networks. In that, the authors proposed the LEACH (Low Energy Adaptive Clustering Hierarchy) algorithm, in which the CH role is dynamically rotated among all sensors in the cluster. Energy is evenly drained from various sensors, leading to improved network lifetime. A similar CH scheduling scheme was proposed in for a time-slotted WSN. In this scheme, several disjoint dominating sets are found and are activated successively. Nodes that are not in the currently active dominating set are put to sleep. A distributed algorithm was proposed to obtain a set-schedule sequence for which the network life time is within a logarithmic factor of the maximum achievable lifetime. In general, rotation based schemes require excessive processing and communication overheads for CH re-election and broadcasting of the new CH information. And moreover the CHs are similar characterised nodes as sensor nodes with same power factor and reliability.

In "**Load-balancing**" algorithms sensors are clustered according to "load-balancing" metrics, whereby the traffic volumes originating from various clusters are equalized, which focus mainly on balancing the intracluster traffic load. The demerit in this algorithm is that the intercluster traffic is not being eliminated in this algorithm and also losses the power in electing the cluster head more.

The Application specific protocol is based on low-energy adaptive clustering hierarchy (LEACH), a protocol architecture for microsensor networks that combines the ideas of energy-efficient cluster-based routing and media access together with application-specific data aggregation to achieve good performance in terms of system lifetime, latency, and application-perceived quality. LEACH includes a new, distributed cluster formation technique that enables self-organization of large numbers of nodes, algorithms for adapting clusters and rotating cluster head positions to evenly distribute the energy load among all the nodes, and techniques to enable distributed signal processing to save communication resources. Our results show that LEACH can improve system lifetime by an order of magnitude compared with general-purpose multi hop approaches.

In the protocol architectures for wireless micro sensor networks, it is important to consider the function of the application, the need for ease of deployment, and the severe energy constraints of the nodes. These features led us to design LEACH, a protocol architecture where computation is performed locally to reduce the amount of transmitted data, network configuration and operation is done using local control, and media access control (MAC) and routing protocols enable low-energy networking.

Topology based clustering

In topology control for wireless sensor networks, a two-tiered Wireless Sensor Network (WSN) consisting of sensor clusters deployed around strategic locations and base-stations (BSs) whose locations are relatively flexible. Within a sensor cluster, there are many small sensor nodes (SNs) that capture, encode and transmit relevant information from the designated area, and there is at least one application node (AN) that receives raw data from these SNs, creates a comprehensive local-view, and forwards the composite bit-stream toward a BS. Both SN and AN are battery-powered and energy- constrained, and their node lifetimes directly affect the network lifetime of WSNs. In-order to maximize the topological network lifetime of the WSN, by arranging BS location and inter-AN relaying optimally. Based on an algorithm in Computational Geometry, we derive the optimal BS locations under three topological lifetime definitions according to mission criticality. When inter-AN relaying becomes feasible and favourable, an optimal parallel relay allocation to further prolong the topological lifetime of the WSN. The experimental performance evaluation demonstrates the efficiency of topology control as a vital process to maximize the network lifetime of WSNs.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE USED:

Processor	: Pentium IV
Processor speed	: 1.5 GHZ
Memory (RAM)	: 3 GB
Hard disk	: 40GB
Monitor	: 15"color monitor
Keyboard	: Logitech 104 keys
Mouse	: Logitech scroll mouse

3.2 SOFTWARE USED:

Virtual machine	: VMware
Operating System	: Red Hat Enterprise LINUX
Language	: TCL & OTcl Scripting
Simulator software	: Network Simulator (NS2)

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. There are different versions of NS and the latest version is ns-2.1b9a while ns-2.1b10 is under development

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

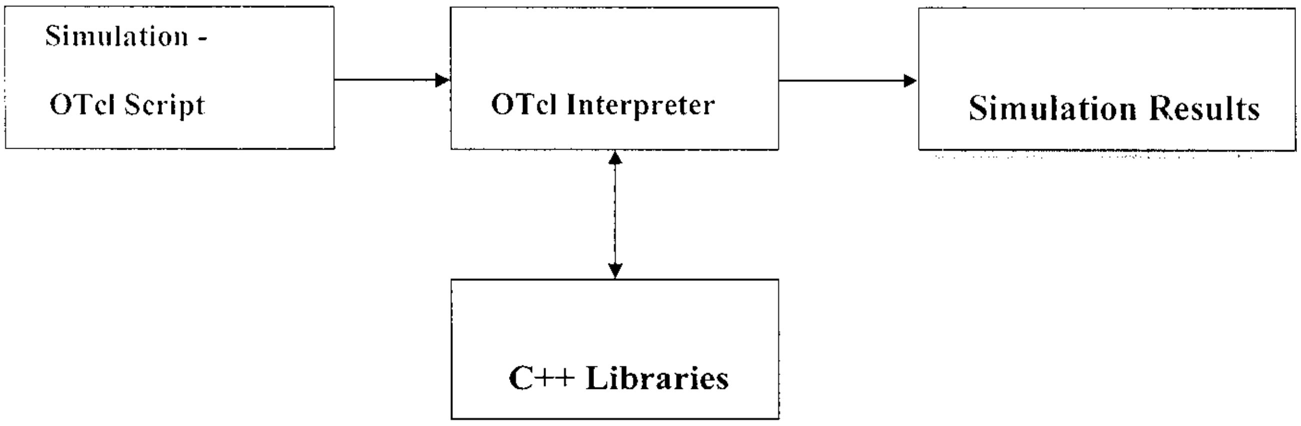


Figure 4.1 Block diagram of Architecture of NS-2

4.3 NETWORK COMPONENTS

This section talks about the NS components, mostly compound network components. Figure 1.1 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, NsObject 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.

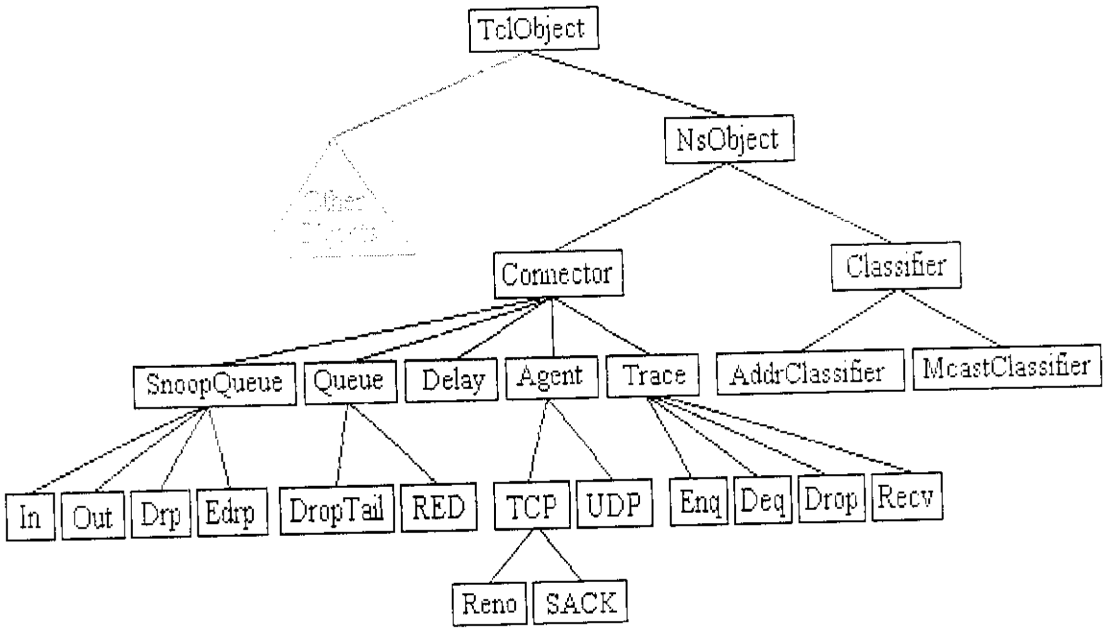


Figure 4.2 OTcl Class Hierarchy

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 Tel 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 Tel& 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. Figure10 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

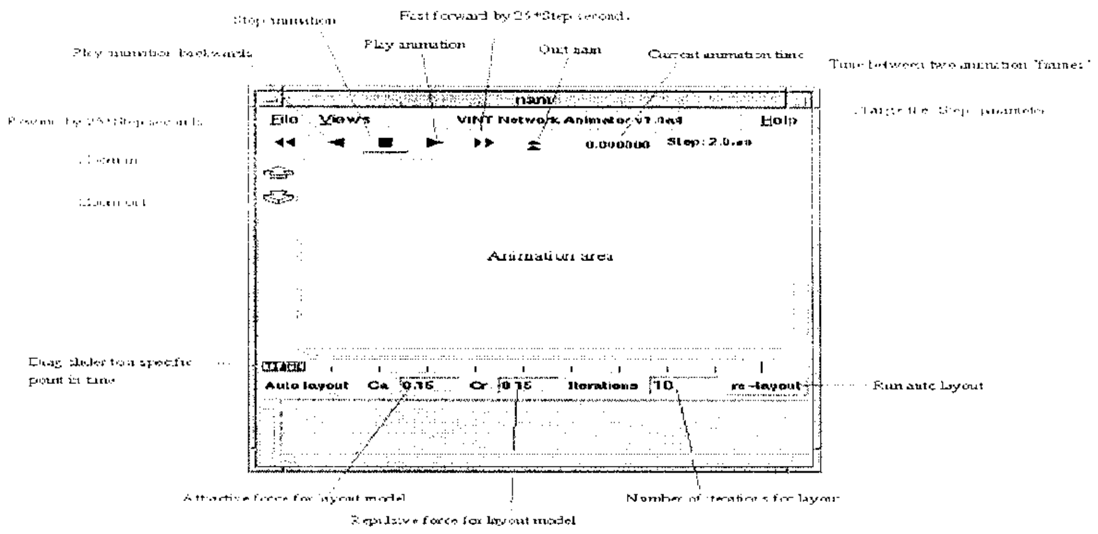
Each trace line starts with an event (+, -, d, r) descriptor followed by the simulation time (in seconds) of that event, and from and to node, which identify the link on which the event occurred information in the line before flags (appeared as "------" since no flag is set) is packet type and size (in 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 and real world packet teaches. 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 11 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 .



Stop animation
Fast forward by 25*Step second.
Play animation backwards
Play animation
Quit nam
Current animation time
Time between two animation "frames"
Rewind by 25*Step seconds
Change the Step parameter
Zoom in
Zoom out
nam
File
Views
VINT Network Animator v1.0a4
Help
0.000000
Step: 2.0ms
Animation area
Drag slider to a specific point in time
Auto layout
Ca
0.15
Cr
0.15
Iterations
10
re-layout
Run auto Layout
Attractive force for layout model
Repulsive force for layout model
Number of iterations for layout

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. This makes energy consumption a critical factor in the design of a WSN and calls for energy-efficient communication protocols that maximize the lifetime of the network. This energy-efficient communication protocol is achieved by investigating the maximization of the coverage time for a clustered wireless sensor network by optimal balancing of power consumption among cluster heads.

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. we investigate the maximization of the coverage time for a clustered wireless sensor network by optimal balancing of power consumption among clusterheads(CHs). Clustering signi?cantly reduces the energy consumption of individual sensors, but it also increases the communication burden on CHs. In the deterministic setup of the nodes, we consider the location of the nodes to be static and the sensor nodes are connected to the Cluster Head and the cluster head connects to the sink. In the stochastic network, the location of the nodes are analyzed as a cone model and the routing between the cluster heads is done. Mechanisms are provided to balance the load in the networks by optimal cluster and routing method in both stochastic and deterministic setup.

5.2 DETERMINISTIC DEPLOYMENT

5.3.1 NETWORK MODEL

We consider a WSN that consists of two types of nodes: Type-I and Type-II nodes. Type-I nodes, which are called sensing nodes (SNs), are responsible for sensing activities. Such nodes are small, low cost, and disposable. They can be densely deployed across the sensing area. Neighbouring SNs are organized in to clusters. A Type-II node has a more powerful energy source and a stronger computing capability and is designated as a CH. Type-II nodes are responsible for receiving and processing the sensing outcomes of SNs. A CH may collect data from intracluster SNs, conduct signal processing on these raw data to create an application-speci?c view of the cluster, and then relay the fused data to the sink through intermediate CHs. Let the numbers of these two types of nodes be M and N, respectively. Suppose that the nodes are arbitrarily placed but their locations are known. No assumptions are made on the shape of the sensing region. The availability of location information is an appropriate assumption in many applications of WSNs. It can also apply to networks where sensors are ?rst randomly deployed but later their locations become known. Each sensing node is assigned to one CH. The sensor generates traffic at an average rate of bits/s and sends it to its CH, which in turn delivers it to the sink. We assume that each sensor has sufficient energy to communicate directly with its CH. This could be done by either transmitting at a high enough transmission power or using a low enough transmission rate and thus a longer duration for each transmitted bit. Furthermore, we assume that the CH depletes its energy at a much faster rate than the sensors it serves. Accordingly, we focus our attention on energy depletion at CHs. From a strategic point of view, CH is more critical to the coverage of the network than individual sensors.

5.3.2 CHANNEL MODEL

We use a Rayleigh fading model to describe the channel between two CHs and also between a CH and the sink. At a transmitter–receiver separation x, the channel gain is given by

$$h(x) = L(d_0)\left(\frac{x}{d_0}\right)^{-n}\xi$$

Where $L(d_0) \overset{\text{def}}{=} \frac{G_t G_r l^2}{16\pi^2 d_0^2}$ is the path loss of the close-in distance d_0. G_t is the antenna gain of the transmitter. G_r is the antenna gain of the receiver. l is the wavelength of the carrier frequency, n is the path loss exponent ($1<n<7$), and *?* is a normalized random variable that represents the ?uctuations in the fading process. Under the assumption of Rayleigh fading, *?* is exponentially distributed.

5.3.3 CLUSTERING SCENARIO

The connection type and the network setup is done. The connection type is setup by the Rayleigh fading channel between the nodes. The deterministic setup provides all the information regarding the location of the nodes in the topology. Thus the sensor nodes are assigned to the cluster heads based on the distance from the nearest cluster head. The number of sensor nodes that are connected to the cluster head are distributed averagely so that no cluster head is overloaded with many sensor nodes than other cluster nodes.

Each cluster head gains the information about its cluster of sensors and maintain the connection between them. In addition, the cluster head also maintains the information of its location with reference to the sink. Using the cluster routing mechanism, the cluster head gathers the information of the path to reach the sink directly if it is near to the sink or through the other cluster head if it is far away from the sink.

While assigning the sensor nodes to the cluster head, it is also noted that it does not exceed the MAX limit of the cluster head.

Algorithm:

```
Input:            $\mathbf{c}^o = (c_1^o, \ldots, c_N^o)$
Initialization:   $\mathbf{U}_1 = \ldots = \mathbf{U}_N = \emptyset$ (cluster sets)
Begin:            For $i = 1$ to $M$
                    For $j = 1$ to $N$
                      set $x_{ij}$ to distance
                      between sensor $i$ and CH $j$
                    endfor
Loop:               $k = \arg_{\{j\}} \min\{x_{ij}, j = 1, \ldots, N\}$
                    if $c_k^o > 0$
                      $c_k^o = c_k^o - \lambda$
                      $\mathbf{U}_k = \mathbf{U}_k + \{k\}$
                    else
                      $x_{ik} = \infty$
                      goto Loop
                    endif
                  endfor
Output:           $\mathbf{U}_1, \ldots, \mathbf{U}_N$
```

5.3.4 ROUTING

The routing between the nodes is carried out using the reactive model routing method. The AODV on-demand routing model is suited for the fast routing deployment without any large use of memory resources unlike the table driven approach.

5.3.5 ENERGY CALCULATION

If a network consist of n number of cluster heads and n+m number of nodes, then the nodes are not assigned in a random order. Each node energy is calculated and their energy needed to communicate the each CH is noted. So the sensor node gets attached to a cluster which requires less energy for communication. This can be directly done by identifying the nearest cluster for it requires less energy to communicate.

If a Cluster Head fails, then the sensor nodes are in search for another cluster head. It is not possible to create a new cluster head or recharge the depleted cluster head immediately. So it is necessary to allocate these sensor nodes to another cluster head. Previous models concentrated only in the need for allotting new cluster head and not in the method of allocation. The nodes that have a failed cluster head is noted. They are in need of a new cluster head. So the cluster head that I closer or the one that has less number of nodes is selected and it is allotted with all the nodes that are noted.

The main disadvantage in this allocation is that the cluster head that is allotted with the extra nodes is used heavily and consumes more power than the other cluster heads. This does not provide an optimal method of reducing the power consumption. So the new method is suppose to be implemented in equally distributing the load of the sensor nodes.

5.3.6 CLUSTER HEAD FAILURE MANAGEMENT

So in this, we provide a unique way for allotting the CHs. By this way, we reduce the burden increase in the other CHs than the previous followed models.

The sensor nodes of the failed Cluster heads search for the nearest cluster head individually. Each sensor nodes identify the nearest cluster head and it is noted. If the selected cluster head has not attained the maximum number of sensor nodes it can monitor, then that particular sensor node is allotted to the cluster head. If not, then the next available nearest cluster head is searched for allotting.

This process repeats until all the un-allotted sensor nodes are allotted. Since a single cluster head is not overloaded with all the sensor nodes of the failed cluster head, the energy consumption is more evenly distributed over the cluster heads than the previous model. The conventional load balancing method is more suitable for the applications of the wireless sensor network. But it is not the optimal solution for managing the cluster head failures.

So the new proposed model manages the cluster head failure situation very well than the conventional method and it is more advantageous and optimize the overall power consumption of the sensor nodes in the wireless sensor network.

CHAPTER 6

EXPERIMENTAL RESULTS AND DISCUSSION

6.1 SIMULATION SCENARIO

The Clustering scenario is based on the distances among the sensor nodes the cluster head is chosen and the data from the sensor node is collected and the data from the cluster head to the nearest cluster heads to reach the destination. This scenario is followed using the AODV protocol and then in case of failure the sensor nodes cannot be replaced. If the cluster head fails need for replacement to transfer the data from the other sensor nodes which sends the data to that particular cluster head by connecting the same sensor node to the nearest cluster head by using the AODV protocol the nearest node is identified and then the data is transferred in case of the area is out of the range it is transferred using the gateway nodes which could be able to reach the specific cluster head.

6.2 PERFORMANCE EVALUATION

The performances analyzed are,

- The transmitting power of the nodes which sends the data to the nearest nodes
- The receiving power of the nodes are being set to a particular value according to the data it may be varying.
- The direct connection of the sensor node to the sink may cause more power consumption and that may lead to the overall network power will be reduced to a large amount.
- The connection between the clusters could be able to reduce the power consumption to transfer the data from the source to the destination.

6.3 PROTOCOL DESCRIPTION

6.3.1 AODV Protocol

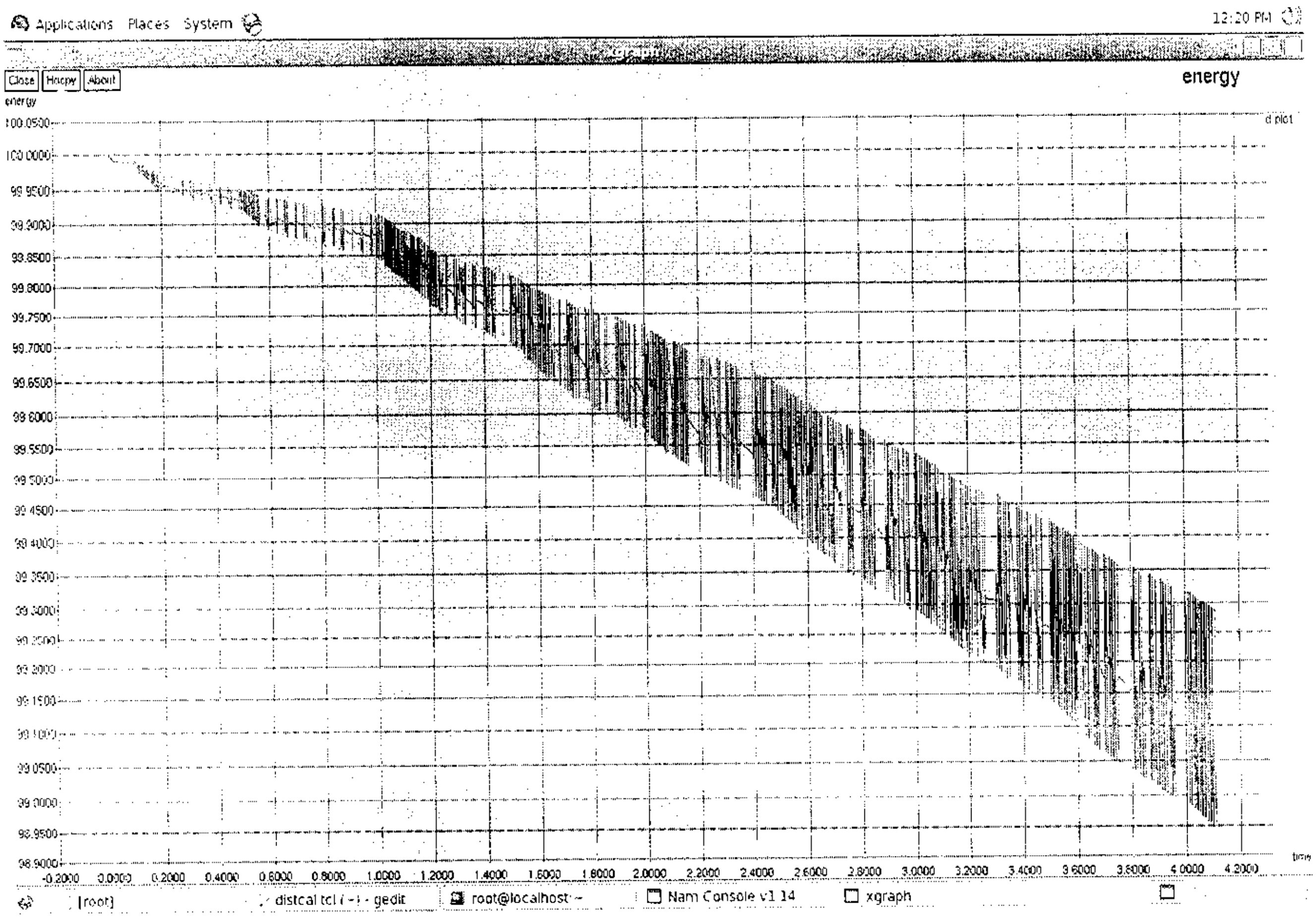
The Ad-hoc On-Demand Distance Vector (AODV) routing protocol is designed for use in ad-hocmobile networks. AODV is a reactive protocol: the routes are created only when they are needed. It uses traditional routing tables, one entry per destination, and sequence numbers to determine whether routing information 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.

6.3.2 Traffic analysis

For clusters with comparable area coverage and node density, the volume of intracluster traffic is roughly the same for all clusters. On the other hand, the traf?c relayed by different CHs is highly skewed; the closer a CH is to the sink, the more traf?c it has to relay, and thus the faster it drains its energy reservoir. Such an imbalanced power consumption situation is essentially caused by the many-to-one communication paradigm in WSNs, i.e., traf?c from all sensors is eventually destined to the sink. If we donot take measures to deliberately balance power consumption at different CHs, a "traf?c implosion" situation will arise. More speci?cally, CHs that are closest to the sink will exhaust their batteries ?rst. Reassigning sensors to the next closest CHs to the sink will simply increase the energy consumption of these CHs. As a result, they will eventually be the second batch of CHs to run out of energy.

This process continues to the next level of CHs, propagating from inside out and eventually leading to early loss of coverage and partitioning of the topology. So in this paper, we propose an efficient method of allocating the sensors of a cluster heads after it runs out of power.

6.4 OPTIMAL ENERGY



6.1 OPTIMAL ENERGY

The above graph shows us the optimal energy consumption of the nodes in the clustered network. The minimal value that is plotted in the graph specifies the remaining energy in the network. This value is much more less than the energy consumed in the un clustered network. So this emphasis on the importance of the clustering scenario in the network.

The main problem that arises in the wireless sensor network is the node failure. And if the failure node is a Cluster head, then it is important to find a replacement with effective optimization of power. After the cluster head failure, the nodes are assigned to the nearest cluster head individually so that no other cluster head is overloaded. This distributes the energy load between the cluster head.

The older model proposes a theory in which the sensor nodes of the failed cluster head is assigned to a single cluster head. This not only increases the load of an individual cluster head, but also the quick drainage of power among the cluster. The below graph specifies the value of the energy remaining in the network following the older model.

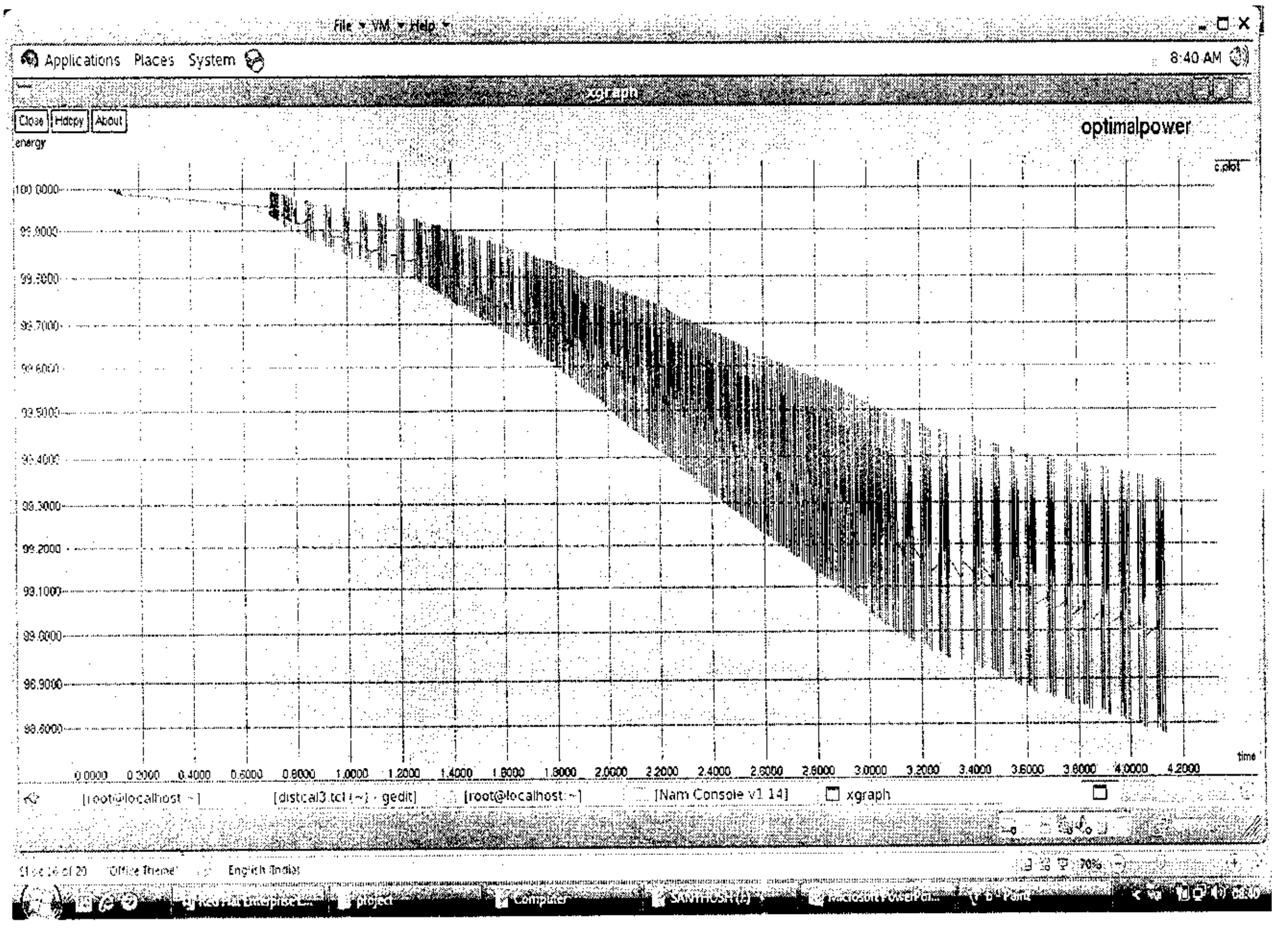


Figure 6.2 Energy Consumption of Older Model using ClusterHead

So we introduce a new method for selective allocation of the sensor nodes of the failed cluster head. Each sensor node belonging to the cluster of the failed cluster head is assigned to its own nearest cluster head that is active. So indirectly, no cluster head is overloaded with maximum number of sensor nodes. The below graph specifies the energy consumption of the network after the node failure in which the improvised method of selective allocation of the sensor nodes is done.

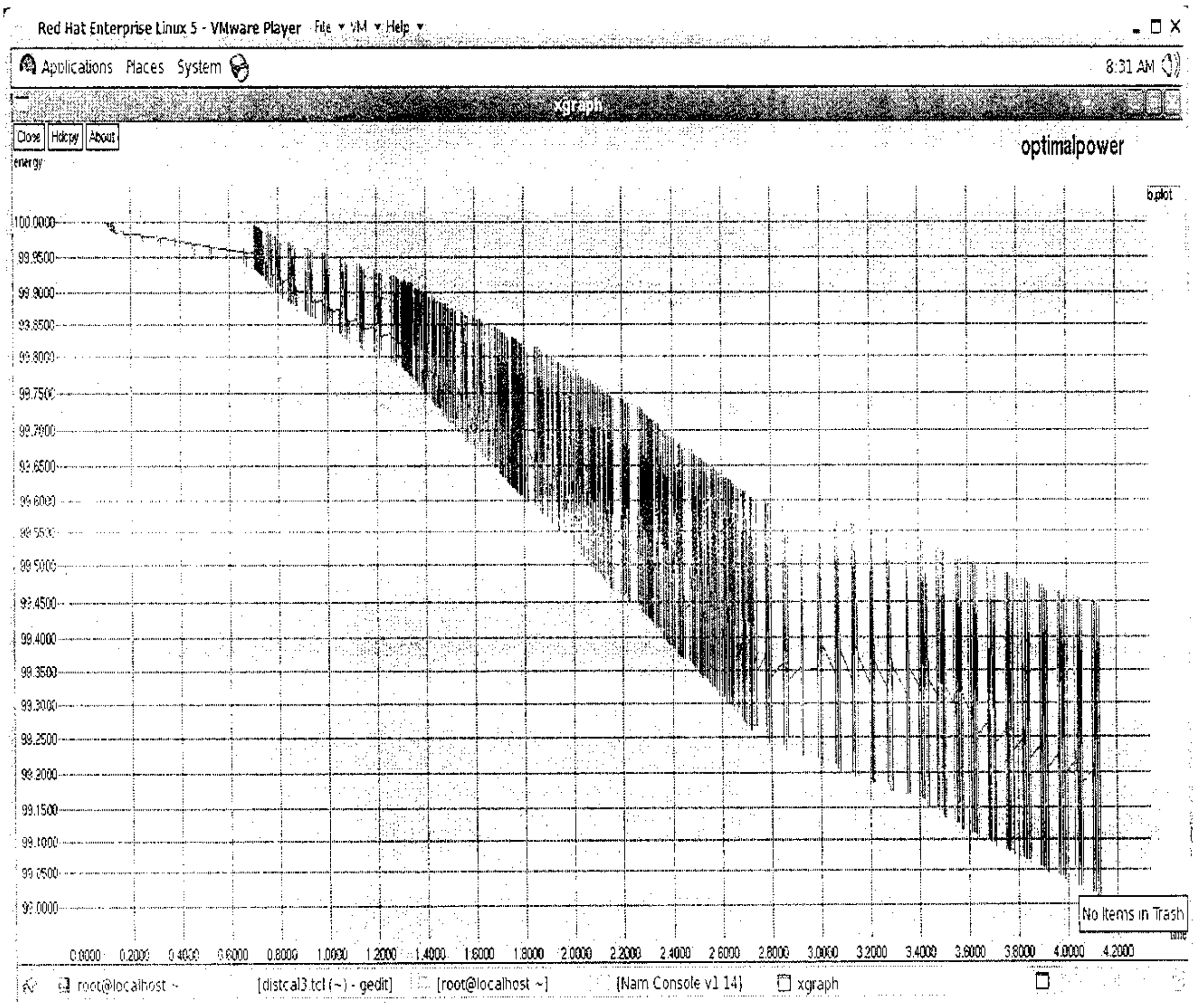


Figure 6.3 Energy Consumption during Failure

COMPARISON RESULT

Comparing the above two immediate graphs, it is strategically clear that the power consumption is more in the conventional method of handling the node failure than our opted model of selective allocation. Thus we provide one of the optimal method to reduce the power consumption in the network without affecting the topology.

This method provides more saving of energy in large networks than smaller networks where more number of clusters are present.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In wireless sensor network, We considered the problem of coverage-time optimization by balancing power consumption at different CHs in a clustered WSN. Stochastic as well as deterministic network models were investigated in our analysis. Our study demonstrates the significance of simultaneously accounting for the impacts of intra and inters cluster traffic in the design of routing and clustering strategies. For the deterministic-topology scenario, we presented a joint clustering/routing optimization based on linear programming. For the stochastic scenario, two mechanisms for balancing power consumption were studied. Simulations were conducted to verify the adequacy of this analysis and demonstrate the substantial bene?t of the proposed schemes in terms of prolonging the coverage time of the network.

(a) The static network can be done with the transfer of data among the nodes to reach the sink in the same cluster.

(b) The dynamic network which moves to different places at times can change the cluster based on the network and the sensor nodes. And that could be able to eliminate the failure of sensor node which cannot transfer the data can be eliminated. The dynamic network which can move and the sensor node can collect the data from all places even though the failure happens.

APPENDIX I

SOURCE CODE

SAMPLE CODE:

```
set sn_num_nodes 4
set si_num_nodes 1
set us_num_nodes 1
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set val(rp) AODV
set val(x) 700
set val(y) 500

set ns [new Simulator]
$ns use-newtrace
set f [open final.tr w]
$ns trace-all $f
set nf [open final.nam w]
$ns namtrace-all-wireless $nf $val(x) $val(y)
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $total_num_nodes

set chan_1_ [new $val(chan)]
$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) \
                -topoInstance $topo \
                -movementTrace OFF \
                -channel $chan_1_

Class Agent/SensorAgent -superclass Agent/MessagePassing

set rng [new RNG]
$rng seed 0
```

```
for {set i 0} {$i <= $sn_num_nodes-1 } {incr i} {
      set SensorNode($i) [$ns node]
      $SensorNode($i) random-motion 0         ;# disable random motion
      set SensorAgent_sn($i) [new Agent/SensorAgent]
      $SensorNode($i) attach  $SensorAgent_sn($i) 42
      $SensorAgent_sn($i) set MyAddr $i
      $SensorAgent_sn($i) set SentBytes 0
      $SensorAgent_sn($i) set RecievedBytes 0
      $SensorNode($i) color red
 }

$SinkNode set X_ 600
$SinkNode set Y_ 300
$SinkNode set Z_ 0.0

$SensorNode(0) set X_ 500
$SensorNode(0) set Y_ 350
$SensorNode(0) set Z_ 0.0

$SensorNode(1) set X_ 450
$SensorNode(1) set Y_ 250
$SensorNode(1) set Z_ 0.0

$SensorNode(2) set X_ 400
$SensorNode(2) set Y_ 325
$SensorNode(2) set Z_ 0.0

for {set i 0} {$i <= $sn_num_nodes-1 } {incr i} {
      $ns at 0.000001 "$SensorNode($i) color red"
}

for {set i 0} {$i <= $sn_num_nodes-1 } {incr i} {
       $ns initial_node_pos $SensorNode($i) 20
}
$ns initial_node_pos $SinkNode 20

$ns initial_node_pos $UserNode 20

$ns at 5.0  "finish"
proc finish {} {
      global ns f nf SensorNode RecievedBytes
      $ns flush-trace
      close $f
        close $nf
      exec nam final.nam &
      exit 0
        }
$ns run
```

Coding:

```
#===================================================================

# Existing system of Wireless Sensor Networks through AODV


#===================================================================

# Define Node Configuration paramaters

#===================================================================


set sn_num_nodes 20
set si_num_nodes 1
set us_num_nodes 1
set total_num_nodes [expr $sn_num_nodes  + [expr $si_num_nodes +
$us_num_nodes]]]
set TotalSimulationTime        10
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set val(rp) AODV
set opt(energymodel)  EnergyModel          ;# Energy model
set opt(radiomodel)   RadioModel           ;# Radio model
set opt(initialenergy)  100                ;# Initial energy in Joules
set val(x) 700
set val(y) 500
set time 10
```

CREATING SIMULATION

```
set ns [new Simulator]
$ns use-newtrace
set f [open route.tr w]
$ns trace-all $f
set nf [open route.nam w]
$ns namtrace-all-wireless $nf $val(x) $val(y)
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $total_num_nodes
```

```
set chan_1_ [new $val(chan)]
$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) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON \
                -movementTrace OFF \
                -initialEnergy $opt(initialenergy) \
                -energyModel $opt(energymodel) \
                -channel $chan_1_

Class Agent/SensorAgent -superclass Agent/MessagePassing

set rng [new RNG]
$rng seed 0

Agent/SensorAgent instproc recv {source sport size data} {
    $self instvar node_ SentBytes RecievedBytes
    if {[$node_ node-addr]==$source} { #receiver's address
        set RecievedBytes [expr $RecievedBytes + $size]
    }
}

Agent/SensorAgent instproc SendSensorData {siv} {
    $self instvar node_
    $self sendto 84 "Data" $siv 42
 }

for {set i 0} {$i <= $sn_num_nodes-1 } {incr i} {
      set SensorNode($i) [$ns node]
      $SensorNode($i) random-motion 0       ;# disable random motion
      set SensorAgent_sn($i) [new Agent/SensorAgent]
      $SensorNode($i) attach  $SensorAgent_sn($i) 42
      $SensorAgent_sn($i) set MyAddr $i
      $SensorAgent_sn($i) set SentBytes 0
      $SensorAgent_sn($i) set RecievedBytes 0
      $SensorNode($i) color red
 }


set SinkNode [$ns node]
```

```
        $SinkNode random-motion 0        ;# disable random motion
        set SensorAgent_si [new Agent/SensorAgent]
        $SinkNode attach  $SensorAgent_si 42
        $SensorAgent_si set MyAddr 0
        $SensorAgent_si set SentBytes 0
        $SensorAgent_si set RecievedBytes 0
        $SinkNode color blue

set UserNode [$ns node]
        $UserNode random-motion 0        ;# disable random motion
        set SensorAgent_us [new Agent/SensorAgent]
        $UserNode attach  $SensorAgent_us 42
        $SensorAgent_us set MyAddr 0
        $SensorAgent_us set SentBytes 0
        $SensorAgent_us set RecievedBytes 0
        $UserNode color green
```

ASSIGNING POSITIONS TO THE NODE

```
$UserNode set X_ 700
$UserNode set Y_ 300
$UserNode set Z_ 0.0


$SinkNode set X_ 600
$SinkNode set Y_ 300
$SinkNode set Z_ 0.0

$SensorNode(0) set X_ 500
$SensorNode(0) set Y_ 350
$SensorNode(0) set Z_ 0.0

$SensorNode(1) set X_ 450
$SensorNode(1) set Y_ 250
$SensorNode(1) set Z_ 0.0

$SensorNode(2) set X_ 400
$SensorNode(2) set Y_ 325
$SensorNode(2) set Z_ 0.0

$SensorNode(3) set X_ 400
$SensorNode(3) set Y_ 200
$SensorNode(3) set Z_ 0.0

$SensorNode(4) set X_ 300
$SensorNode(4) set Y_ 400
$SensorNode(4) set Z_ 0.0
```

```
$SensorNode(5) set X_ 200
$SensorNode(5) set Y_ 300
$SensorNode(5) set Z_ 0.0

$SensorNode(6) set X_ 200
$SensorNode(6) set Y_ 400
$SensorNode(6) set Z_ 0.0

$SensorNode(7) set X_ 350
$SensorNode(7) set Y_ 500
$SensorNode(7) set Z_ 0.0

$SensorNode(8) set X_ 150
$SensorNode(8) set Y_ 525
$SensorNode(8) set Z_ 0.0

$SensorNode(9) set X_ 0
$SensorNode(9) set Y_ 450
$SensorNode(9) set Z_ 0.0

$SensorNode(10) set X_ 30
$SensorNode(10) set Y_ 400
$SensorNode(10) set Z_ 0.0

$SensorNode(11) set X_ 25
$SensorNode(11) set Y_ 550
$SensorNode(11) set Z_ 0.0

$SensorNode(12) set X_ 100
$SensorNode(12) set Y_ 200
$SensorNode(12) set Z_ 0.0

$SensorNode(13) set X_ 0
$SensorNode(13) set Y_ 150
$SensorNode(13) set Z_ 0.0

$SensorNode(14) set X_ 125
$SensorNode(14) set Y_ 125
$SensorNode(14) set Z_ 0.0

$SensorNode(15) set X_ 20
$SensorNode(15) set Y_ 50
$SensorNode(15) set Z_ 0.0

$SensorNode(16) set X_ 450
$SensorNode(16) set Y_ 300
$SensorNode(16) set Z_ 0.0
```

```
$SensorNode(17) set X_ 250
$SensorNode(17) set Y_ 350
$SensorNode(17) set Z_ 0.0

$SensorNode(18) set X_ 75
$SensorNode(18) set Y_ 475
$SensorNode(18) set Z_ 0.0

$SensorNode(19) set X_ 50
$SensorNode(19) set Y_ 100
$SensorNode(19) set Z_ 0.0
```

```
for {set i 0} {$i <= $sn_num_nodes-1 } {incr i} {
     $ns at 0.000001 "$SensorNode($i) color red"
}


$ns at 0.000001 "$SinkNode color blue"
$ns at 0.000001 "$UserNode color green"

$ns at 0 "$SinkNode label Sink"
$ns at 0 "$UserNode label User"

for {set i 0} {$i <= $sn_num_nodes-1 } {incr i} {
       $ns initial_node_pos $SensorNode($i) 20
}
```

CONNECTION AMONG THE NODES

```
set udp0 [$ns create-connection UDP $SensorNode(16) LossMonitor
$SinkNode 0]
      $udp0 set fid_ 1
      set cbr0 [$udp0 attach-app Traffic/CBR]
      $cbr0 set packetSize_ 1000
      $cbr0 set interval_ .07
      $ns at 0.1 "$cbr0 start"
      $ns at 4.1 "$cbr0 stop"

set udp1 [$ns create-connection UDP $SensorNode(19) LossMonitor
$SinkNode 0]
      $udp1 set fid_ 1
      set cbr1 [$udp1 attach-app Traffic/CBR]
      $cbr1 set packetSize_ 1000
      $cbr1 set interval_ .07
      $ns at 0.1 "$cbr1 start"
      $ns at 4.1 "$cbr1 stop"
```

```
set udp2 [$ns create-connection UDP $SensorNode(11) LossMonitor
$SinkNode 0]
      $udp2 set fid_ 1
      set cbr2 [$udp2 attach-app Traffic/CBR]
      $cbr2 set packetSize_ 1000
      $cbr2 set interval_ .07
      $ns at 0.1 "$cbr2 start"
      $ns at 2.1 "$cbr2 stop"

set udp3 [$ns create-connection UDP $SensorNode(9) LossMonitor
$SinkNode 0]
      $udp3 set fid_ 1
      set cbr3 [$udp3 attach-app Traffic/CBR]
      $cbr3 set packetSize_ 1000
      $cbr3 set interval_ .07
      $ns at 0.1 "$cbr3 start"
      $ns at 2.1 "$cbr3 stop"
set udp4 [$ns create-connection UDP $SensorNode(15) LossMonitor
$SinkNode 0]
      $udp4 set fid_ 1
      set cbr4 [$udp4 attach-app Traffic/CBR]
      $cbr4 set packetSize_ 1000
      $cbr4 set interval_ .07
      $ns at 0.1 "$cbr4 start"
      $ns at 2.1 "$cbr4 stop"
#for {set i 0} {$i <= $ch_num_nodes-1 } {incr i} {
 #      $ns initial_node_pos $ClusterHead($i) 20
#}


$ns initial_node_pos $SinkNode 20

$ns initial_node_pos $UserNode 20

$ns at 4.1  "finish"
proc finish {} {
      global ns f nf SensorNode RecievedBytes
      $ns flush-trace
      close $f
        close $nf
      exec nam route.nam &
      exit 0
       }
$ns run
```

```
#===================================================================

# Implemented clustered Wireless Sensor Networks through AODV

#===================================================================

# Define Node Configuration paramaters

#===================================================================

set sn_num_nodes 16
set ch_num_nodes 4
set si_num_nodes 1
set us_num_nodes 1
set total_num_nodes [expr $sn_num_nodes + [expr $ch_num_nodes + [expr
$si_num_nodes + $us_num_nodes]]]
set TotalSimulationTime        10
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set opt(energymodel)  EnergyModel         ;# Energy model
set opt(radiomodel)   RadioModel          ;# Radio model
set opt(initialenergy)  100               ;# Initial energy in Joules
set val(rp) AODV
set val(x) 700
set val(y) 500
#set opt(initialenergy)  100
set time 10



set ns [new Simulator]
$ns use-newtrace
set f [open final.tr w]
$ns trace-all $f
set nf [open final.nam w]
$ns namtrace-all-wireless $nf $val(x) $val(y)
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $total_num_nodes
```

```
set chan_1_ [new $val(chan)]
$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) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON \
                -movementTrace OFF \
            -initialEnergy $opt(initialenergy) \
                -energyModel $opt(energymodel) \
                -channel $chan_1_


Class Agent/SensorAgent -superclass Agent/MessagePassing

set rng [new RNG]
$rng seed 0

Agent/SensorAgent instproc recv {source sport size data} {
    $self instvar node_ SentBytes RecievedBytes
    if {[$node_ node-addr]==$source} { #receiver's address
        set RecievedBytes [expr $RecievedBytes + $size]
    }
}

Agent/SensorAgent instproc SendSensorData {siv} {
    $self instvar node_
    $self sendto 84 "Data" $siv 42
 }



for {set i 0} {$i <= $sn_num_nodes-1 } {incr i} {
      set SensorNode($i) [$ns node]
      $SensorNode($i) random-motion 0        ;# disable random motion
      set SensorAgent_sn($i) [new Agent/SensorAgent]
      $SensorNode($i) attach  $SensorAgent_sn($i) 42
      $SensorAgent_sn($i) set MyAddr $i
      $SensorAgent_sn($i) set SentBytes 0
      $SensorAgent_sn($i) set RecievedBytes 0
```

```
        $SensorNode($i) color red
    }

  for {set i 0} {$i <= $ch_num_nodes-1 } {incr i} {
        set ClusterHead($i) [$ns node]
        $ClusterHead($i) random-motion 0      ;# disable random motion
        set SensorAgent_ch($i) [new Agent/SensorAgent]
        $ClusterHead($i) attach  $SensorAgent_ch($i) 42
        $SensorAgent_ch($i) set MyAddr $i
        $SensorAgent_ch($i) set SentBytes 0
        $SensorAgent_ch($i) set RecievedBytes 0
        $ClusterHead($i) color yellow
  }

  set SinkNode [$ns node]
        $SinkNode random-motion 0       ;# disable random motion
        set SensorAgent_si [new Agent/SensorAgent]
        $SinkNode attach  $SensorAgent_si 42
        $SensorAgent_si set MyAddr 0
        $SensorAgent_si set SentBytes 0
        $SensorAgent_si set RecievedBytes 0
        $SinkNode color blue

  set UserNode [$ns node]
        $UserNode random-motion 0       ;# disable random motion
        set SensorAgent_us [new Agent/SensorAgent]
        $UserNode attach  $SensorAgent_us 42
        $SensorAgent_us set MyAddr 0
        $SensorAgent_us set SentBytes 0
        $SensorAgent_us set RecievedBytes 0
        $UserNode color green



  $UserNode set X_ 700
  $UserNode set Y_ 300
  $UserNode set Z_ 0.0


  $SinkNode set X_ 600
  $SinkNode set Y_ 300
  $SinkNode set Z_ 0.0

  $SensorNode(0) set X_ 500
  $SensorNode(0) set Y_ 350
  $SensorNode(0) set Z_ 0.0

  $SensorNode(1) set X_ 450
  $SensorNode(1) set Y_ 250
```

```
$SensorNode(1) set Z_ 0.0

$SensorNode(2) set X_ 400
$SensorNode(2) set Y_ 325
$SensorNode(2) set Z_ 0.0

$SensorNode(3) set X_ 400
$SensorNode(3) set Y_ 200
$SensorNode(3) set Z_ 0.0

$SensorNode(4) set X_ 300
$SensorNode(4) set Y_ 400
$SensorNode(4) set Z_ 0.0

$SensorNode(5) set X_ 200
$SensorNode(5) set Y_ 300
$SensorNode(5) set Z_ 0.0

$SensorNode(6) set X_ 200
$SensorNode(6) set Y_ 400
$SensorNode(6) set Z_ 0.0

$SensorNode(7) set X_ 350
$SensorNode(7) set Y_ 500
$SensorNode(7) set Z_ 0.0

$SensorNode(8) set X_ 150
$SensorNode(8) set Y_ 525
$SensorNode(8) set Z_ 0.0

$SensorNode(9) set X_ 0
$SensorNode(9) set Y_ 450
$SensorNode(9) set Z_ 0.0

$SensorNode(10) set X_ 30
$SensorNode(10) set Y_ 400
$SensorNode(10) set Z_ 0.0

$SensorNode(11) set X_ 20
$SensorNode(11) set Y_ 550
$SensorNode(11) set Z_ 0.0

$SensorNode(12) set X_ 100
$SensorNode(12) set Y_ 200
$SensorNode(12) set Z_ 0.0

$SensorNode(13) set X_ 0
$SensorNode(13) set Y_ 150
$SensorNode(13) set Z_ 0.0
```

```
$SensorNode(14) set X_ 125
$SensorNode(14) set Y_ 125
$SensorNode(14) set Z_ 0.0


$SensorNode(15) set X_ 20
$SensorNode(15) set Y_ 50
$SensorNode(15) set Z_ 0.0

$ClusterHead(0) set X_ 450
$ClusterHead(0) set Y_ 300
$ClusterHead(0) set Z_ 0.0

$ClusterHead(1) set X_ 250
$ClusterHead(1) set Y_ 350
$ClusterHead(1) set Z_ 0.0

$ClusterHead(2) set X_ 75
$ClusterHead(2) set Y_ 475
$ClusterHead(2) set Z_ 0.0

$ClusterHead(3) set X_ 50
$ClusterHead(3) set Y_ 100
$ClusterHead(3) set Z_ 0.0
```

Distance Calculation

```
proc distance {X1 Y1 X2 Y2} {
set z [expr pow([expr pow([expr $X2 - $X1],2)+pow([expr $Y2 -
$Y1],2)],0.5)]
[return $z]
}


                for {set i 0} {$i <= $sn_num_nodes-1} {incr i} {
                    for {set j 0} {$j <= $ch_num_nodes-1} {incr j} {
                        set sn_ch_dis($i,$j) [distance [$SensorNode($i)
set X_] [$SensorNode($i) set Y_] [$ClusterHead($j) set X_]
[$ClusterHead($j) set Y_]]
puts "[$SensorNode($i) set X_] : [$SensorNode($i) set Y_] ::
[$ClusterHead($j) set X_] : [$ClusterHead($j) set
Y_]==>$sn_ch_dis($i,$j)"
                        }
                    }

puts ""
```

```
 for {set i 0} {$i<$ch_num_nodes} {incr i} {
            for {set j 0} {$j<$ch_num_nodes} {incr j} {
                  set ch_ch_si_dis($i,$j) [distance [$ClusterHead($i)
set X_] [$ClusterHead($i) set Y_] [$ClusterHead($j) set X_]
[$ClusterHead($j) set Y_]]
                  if {$ch_ch_si_dis($i,$j)==0} { set
ch_ch_si_dis($i,$j) 1000.0 }

puts "[$ClusterHead($i) set X_] : [$ClusterHead($i) set Y_] ::
[$ClusterHead($j) set X_] : [$ClusterHead($j) set
Y_]==>$ch_ch_si_dis($i,$j)"

                  if {$i==$ch_num_nodes-1 && $j==$ch_num_nodes-1} {
                        set ch_ch_si_dis($i+1,$j+1) [distance
[$SinkNode set X_] [$SinkNode set Y_] [$SinkNode set X_] [$SinkNode
set Y_]]
                        if {$ch_ch_si_dis($i+1,$j+1)==0} { set
ch_ch_si_dis($i+1,$j+1) 1000.0 }
puts "[$SinkNode set X_] si [$SinkNode set Y_] :: [$SinkNode set X_]
si [$SinkNode set Y_]==>$ch_ch_si_dis($i+1,$j+1)"
                  }
                  if {$i==$ch_num_nodes-1} {
                        set ch_ch_si_dis($i+1,$j) [distance
[$SinkNode set X_] [$SinkNode set Y_] [$ClusterHead($j) set X_]
[$ClusterHead($j) set Y_]]
                        if {$ch_ch_si_dis($i+1,$j)==0} { set
ch_ch_si_dis($i+1,$j) 1000.0 }
puts "[$SinkNode set X_] si [$SinkNode set Y_] :: [$ClusterHead($j)
set X_] : [$ClusterHead($j) set Y_]==>$ch_ch_si_dis($i+1,$j)"
                  }
                  if {$j==$ch_num_nodes-1} {
                        set ch_ch_si_dis($i,$j+1) [distance
[$ClusterHead($i) set X_] [$ClusterHead($i) set Y_] [$SinkNode set X_]
[$SinkNode set Y_]]
                        if {$ch_ch_si_dis($i,$j+1)==0} { set
ch_ch_si_dis($i,$j+1) 1000.0 }
puts "[$ClusterHead($i) set X_] : [$ClusterHead($i) set Y_] ::
[$SinkNode set X_] si [$SinkNode set Y_]==>$ch_ch_si_dis($i,$j+1)"
                  }
 }
 }

 ######################################################################
 ############
 for {set i 0} {$i <= $sn_num_nodes-1 } {incr i} {
      $ns at 0.000001 "$SensorNode($i) color red"
 }
```

```
for {set i 0} {$i <= $ch_num_nodes-1 } {incr i} {
    $ns at 0.000001 "$ClusterHead($i) color yellow"
}

$ns at 0.000001 "$SinkNode color blue"
$ns at 0.000001 "$UserNode color green"

$ns at 0 "$SinkNode label Sink"
$ns at 0 "$UserNode label User"

for {set i 0} {$i <= $sn_num_nodes-1 } {incr i} {
      $ns initial_node_pos $SensorNode($i) 20
}

for {set i 0} {$i <= $ch_num_nodes-1 } {incr i} {
      $ns initial_node_pos $ClusterHead($i) 20
}

$ns initial_node_pos $SinkNode 20

$ns initial_node_pos $UserNode 20
#coloring nodes
$SensorNode(0) color black
$ns at 2.1 "$SensorNode(0) color black"

#establishing connection

set udp0 [$ns create-connection UDP $SensorNode(0) LossMonitor
$ClusterHead(0) 0]
      $udp0 set fid_ 1
      set cbr0 [$udp0 attach-app Traffic/CBR]
      $cbr0 set packetSize_ 100
      $cbr0 set interval_ .07
      $ns at 0.0 "$cbr0 start"
      $ns at 2.1 "$cbr0 stop"

set udp1 [$ns create-connection UDP $SensorNode(1) LossMonitor
$ClusterHead(0) 0]
      $udp1 set fid_ 1
      set cbr1 [$udp1 attach-app Traffic/CBR]
      $cbr1 set packetSize_ 1000
      $cbr1 set interval_ .07
      $ns at 0.1 "$cbr1 start"
      $ns at 4.1 "$cbr1 stop"

set udp2 [$ns create-connection UDP $SensorNode(2) LossMonitor
$ClusterHead(0) 0]
      $udp2 set fid_ 1
```

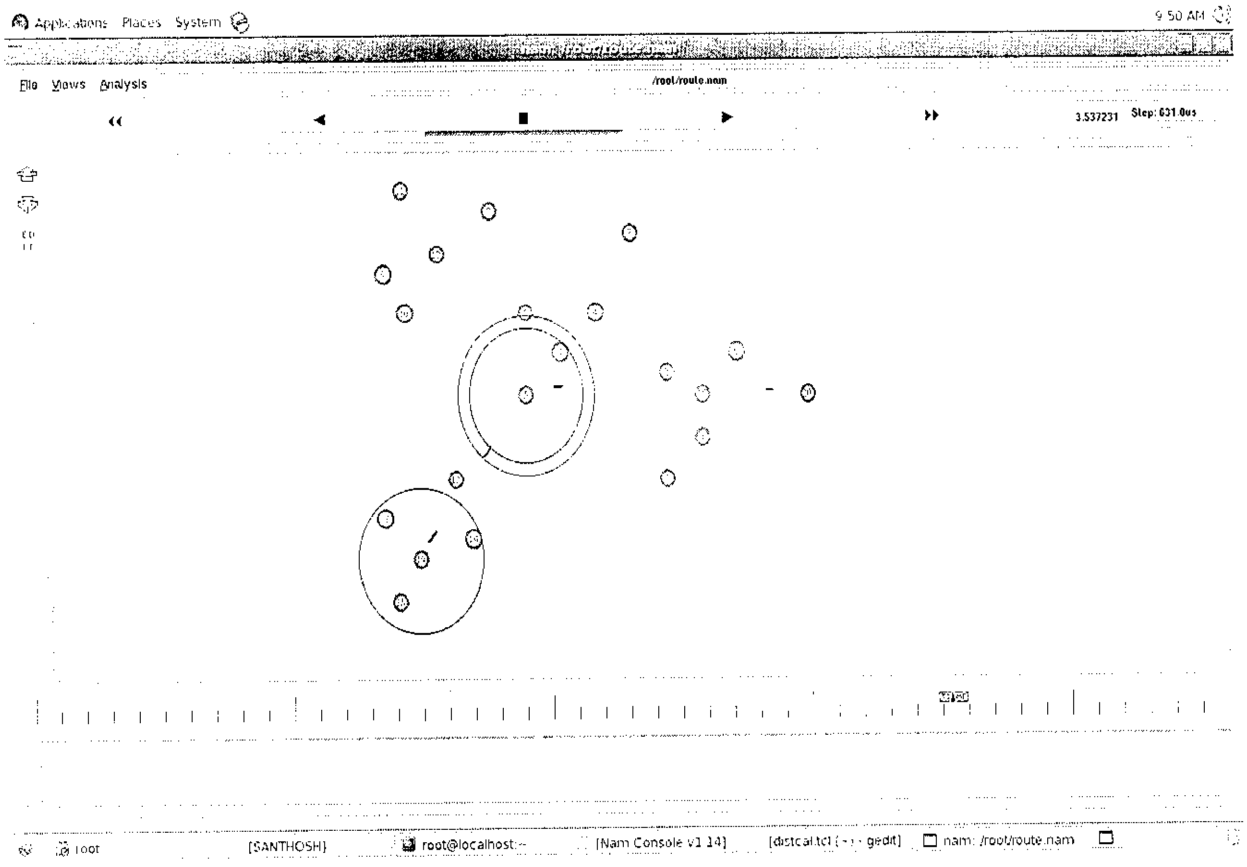
```
        set cbr2 [$udp2 attach-app Traffic/CBR]
        $cbr2 set packetSize_ 1000
        $cbr2 set interval_ .07
        $ns at 2.4 "$cbr2 start"
        $ns at 4.1 "$cbr2 stop"

set udp3 [$ns create-connection UDP $SensorNode(5) LossMonitor
$ClusterHead(1) 0]
        $udp3 set fid_ 1
        set cbr3 [$udp3 attach-app Traffic/CBR]
        $cbr3 set packetSize_ 1000
        $cbr3 set interval_ .1
        $ns at 1.0 "$cbr3 start"
        $ns at 4.1 "$cbr3 stop"

set udp4 [$ns create-connection UDP $ClusterHead(1) LossMonitor
$ClusterHead(0) 0]
        $udp4 set fid_ 1
        set cbr4 [$udp4 attach-app Traffic/CBR]
        $cbr4 set packetSize_ 1000
        $cbr4 set interval_ .5
        $ns at 1.0 "$cbr4 start"
        $ns at 4.1 "$cbr4 stop"

set udp5 [$ns create-connection UDP $ClusterHead(0) LossMonitor
$SinkNode 0]
        $udp5 set fid_ 1
        set cbr5 [$udp5 attach-app Traffic/CBR]
        $cbr5 set packetSize_ 1000
        $cbr5 set interval_ .5
        $ns at 1.1 "$cbr5 start"
        $ns at 4.1 "$cbr5 stop"

set udp6 [$ns create-connection UDP $ClusterHead(3) LossMonitor
$ClusterHead(1) 0]
        $udp6 set fid_ 1
        set cbr6 [$udp6 attach-app Traffic/CBR]
        $cbr6 set packetSize_ 1000
        $cbr6 set interval_ .5
        $ns at 0.5 "$cbr6 start"
        $ns at 4.1 "$cbr6 stop"

set udp7 [$ns create-connection UDP $SensorNode(8) LossMonitor
$ClusterHead(2) 0]
        $udp7 set fid_ 1
        set cbr7 [$udp7 attach-app Traffic/CBR]
        $cbr7 set packetSize_ 1000
        $cbr7 set interval_ .1
        $ns at 1.1 "$cbr7 start"
```

```
      $ns at 4.1 "$cbr7 stop"

set udp8 [$ns create-connection UDP $SensorNode(12) LossMonitor
$ClusterHead(3) 0]
      $udp8 set fid_ 1
      set cbr8 [$udp8 attach-app Traffic/CBR]
      $cbr8 set packetSize_ 1000
      $cbr8 set interval_ .5
      $ns at 0.1 "$cbr8 start"
      $ns at 4.1 "$cbr8 stop"


#annotation deatils
$ns at 2.15 "$ns trace-annotate \"SensorNode  0  Failed \" "


$ns at 5.0  "finish"

proc finish {} {
      global ns f nf SensorNode RecievedBytes
      $ns flush-trace
      close $f
        close $nf
      exec nam final.nam &


      exit 0
        }
$ns run
```

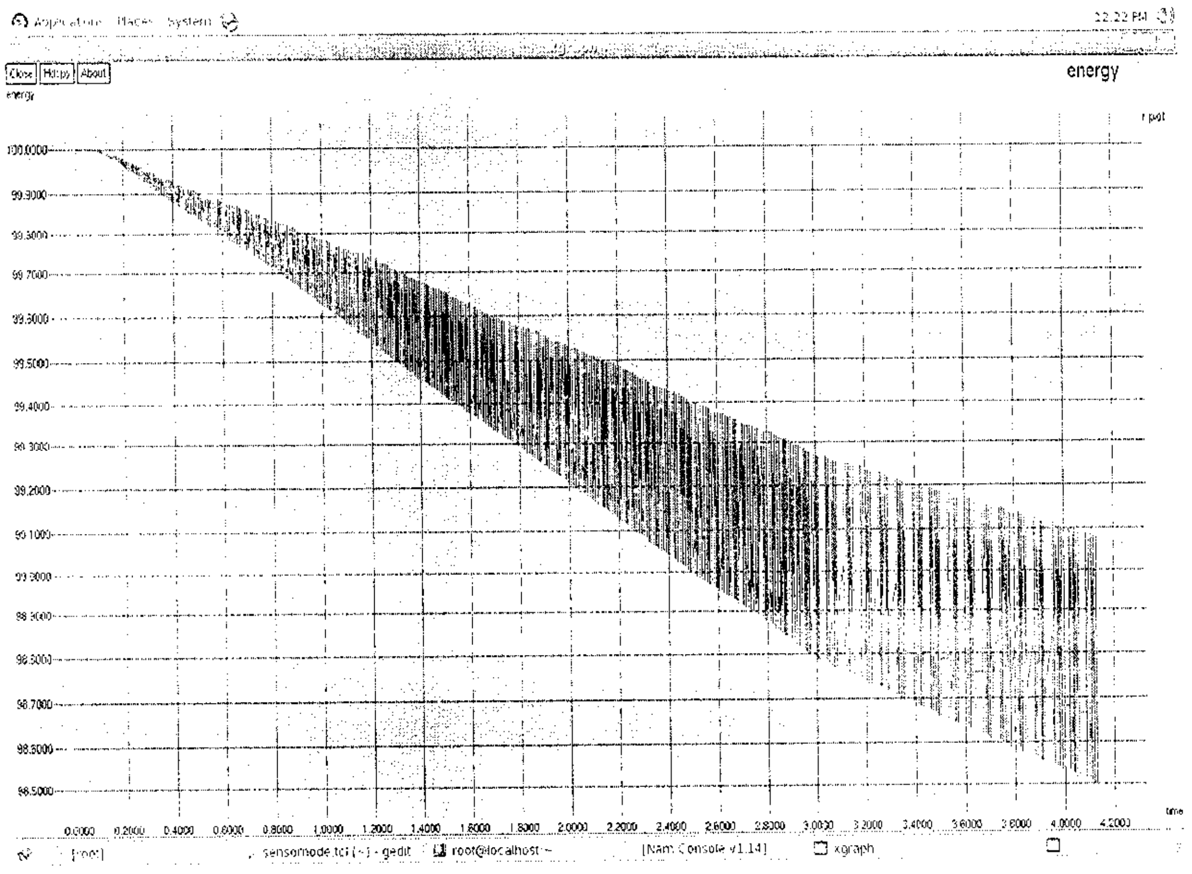
APPENDIX II

SNAPSHOT

TRANSFER OF DATA USING SENSORNODES WITHOUT CLUSTERING

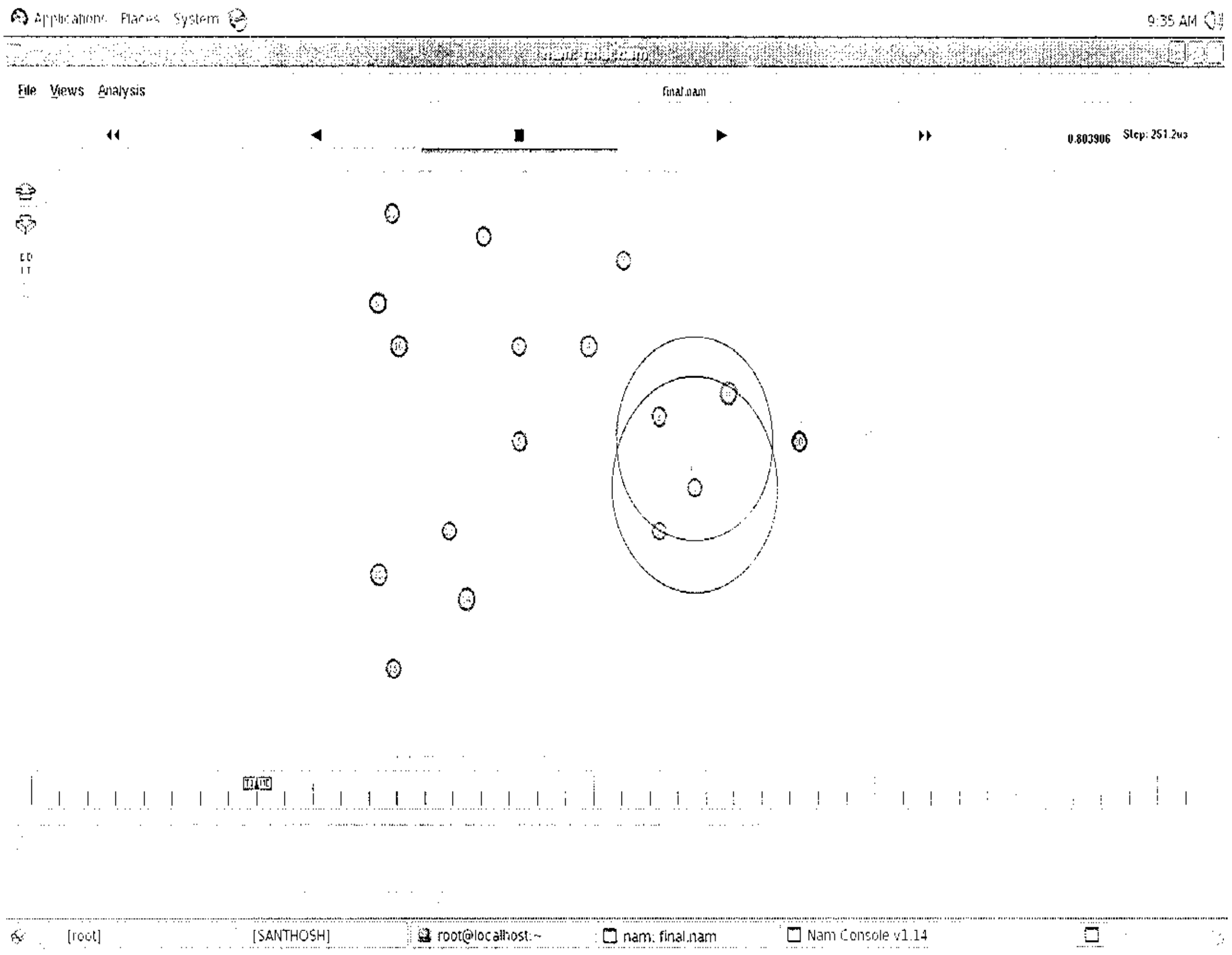


POWER CONSUMPTION FOR A WIRELESS NETWORK (WITHOUT CLUSTERING)

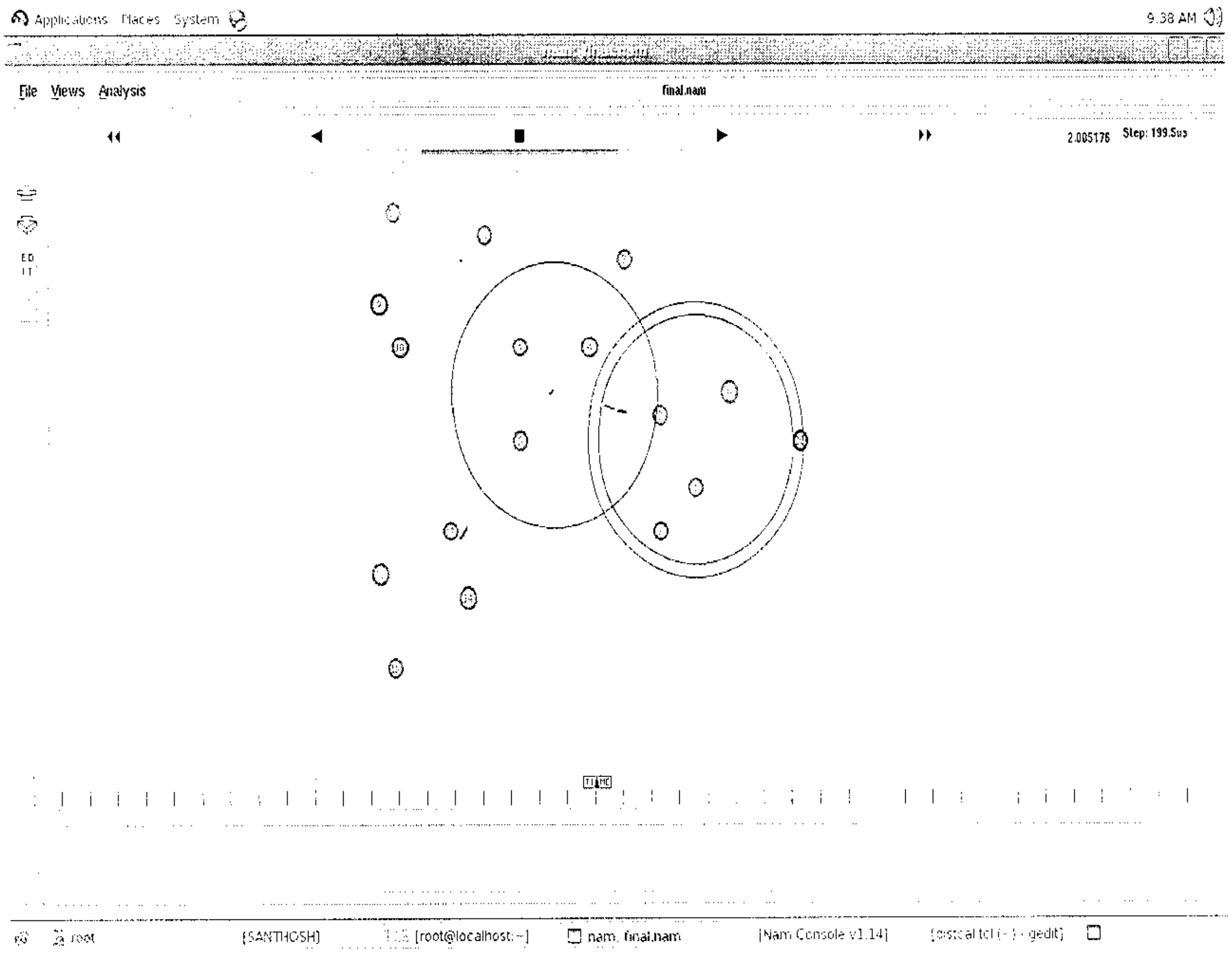


IMPLEMENTED SCENARIO:

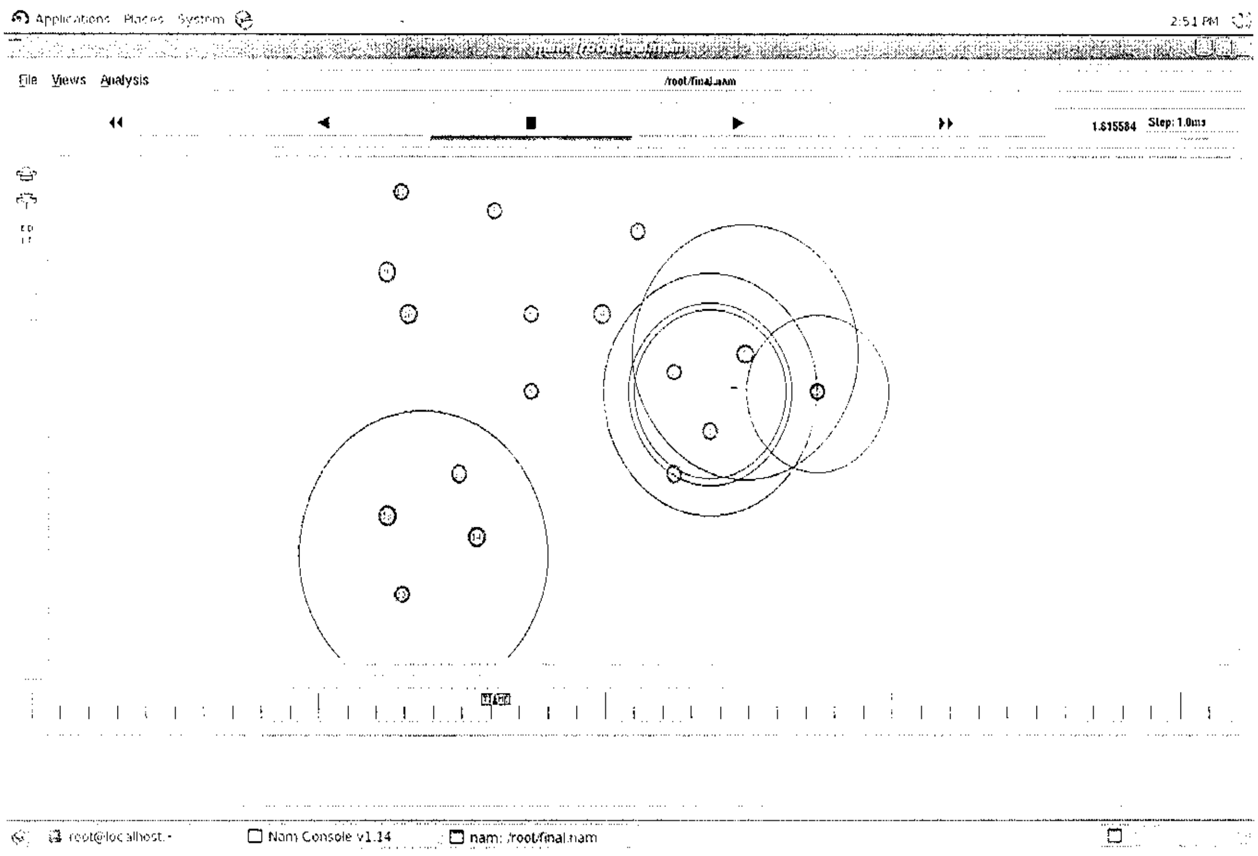
TRANSFER OF DATA AMONG NODES WITH CLUSTERING



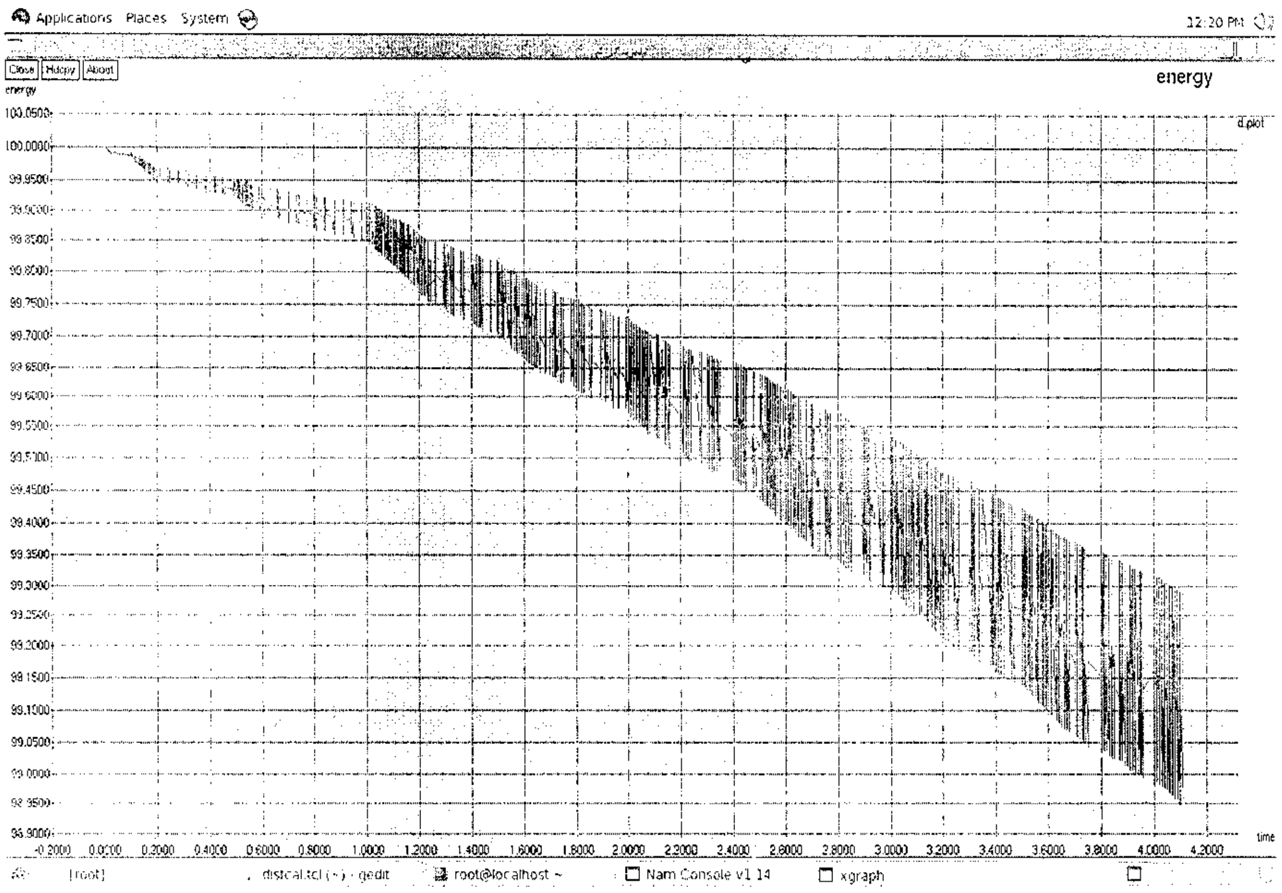
DATA FORWARDING FROM THE CLUSTERHEAD TO NEARER CLUSTERHEAD TO REACH SINK



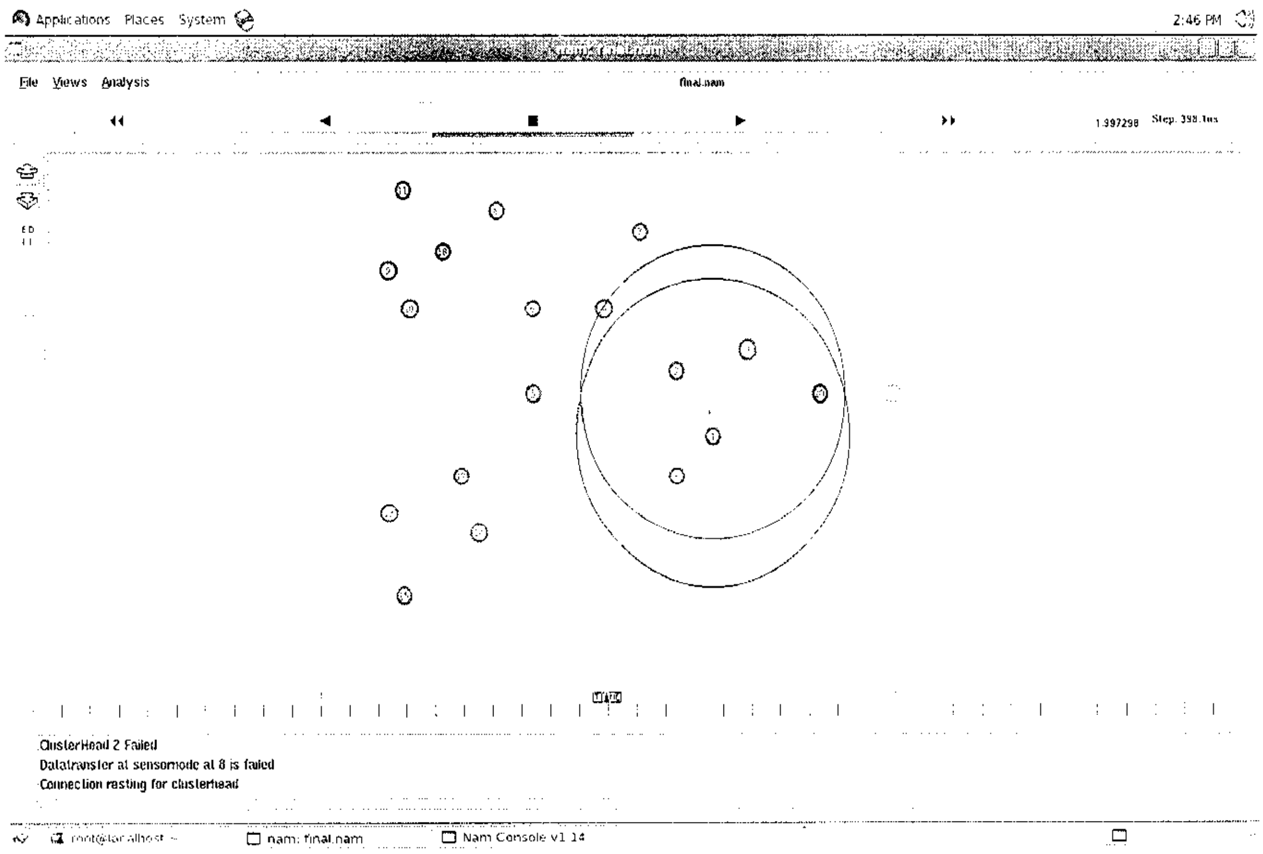
DATA FROM CLUSTERHEAD TO SINK



OPTIMAL POWER CONSUMPTION OF THE WIRELESS NETWORK (WITH CLUSTERING)

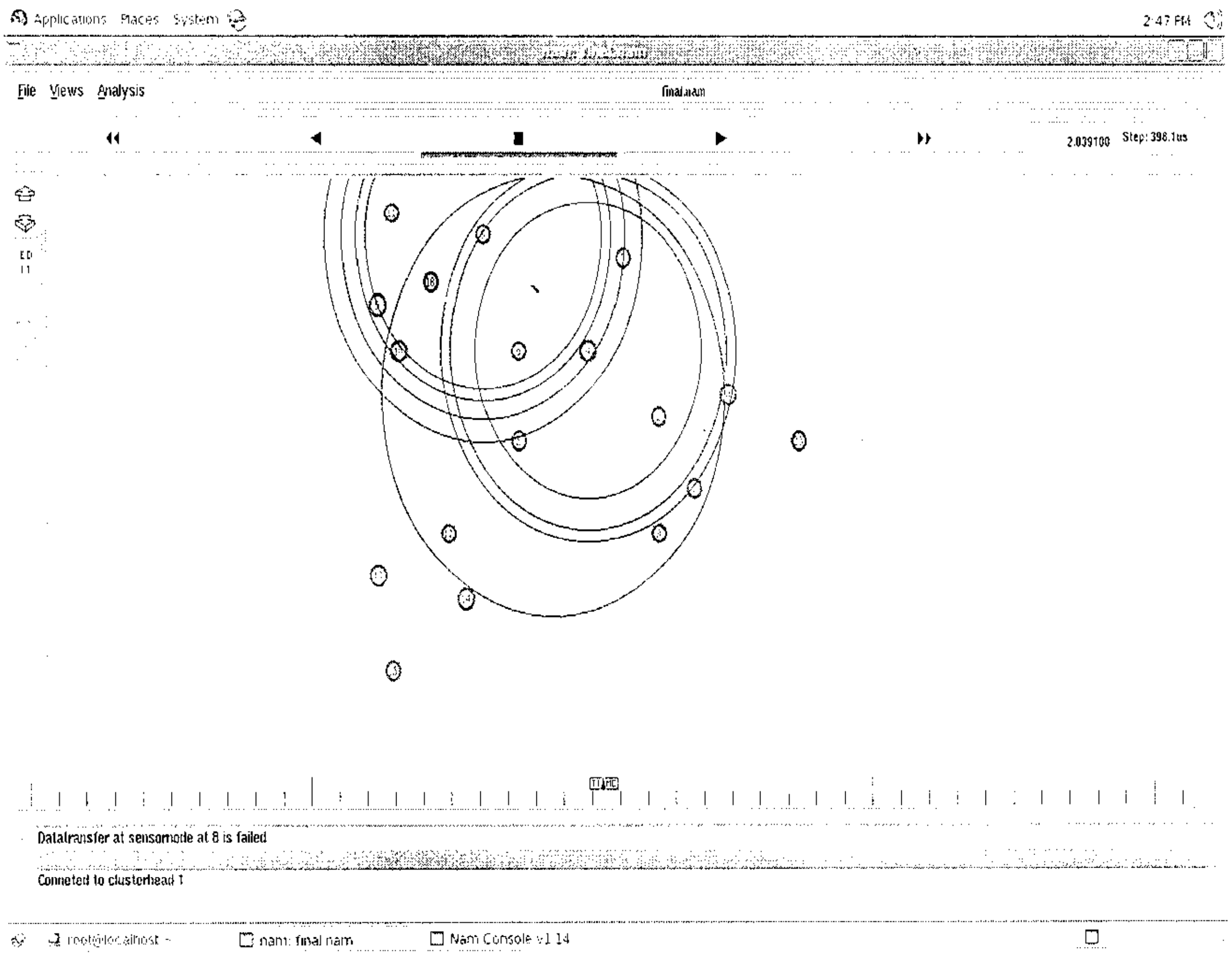


CLUSTERHEAD FAILURE



```
ClusterHead 2 Failed
Datatransfer at sensornode at 8 is failed
Connection resting for clusterhead
```

CONNECTION RESETTING WHILE FAILURE

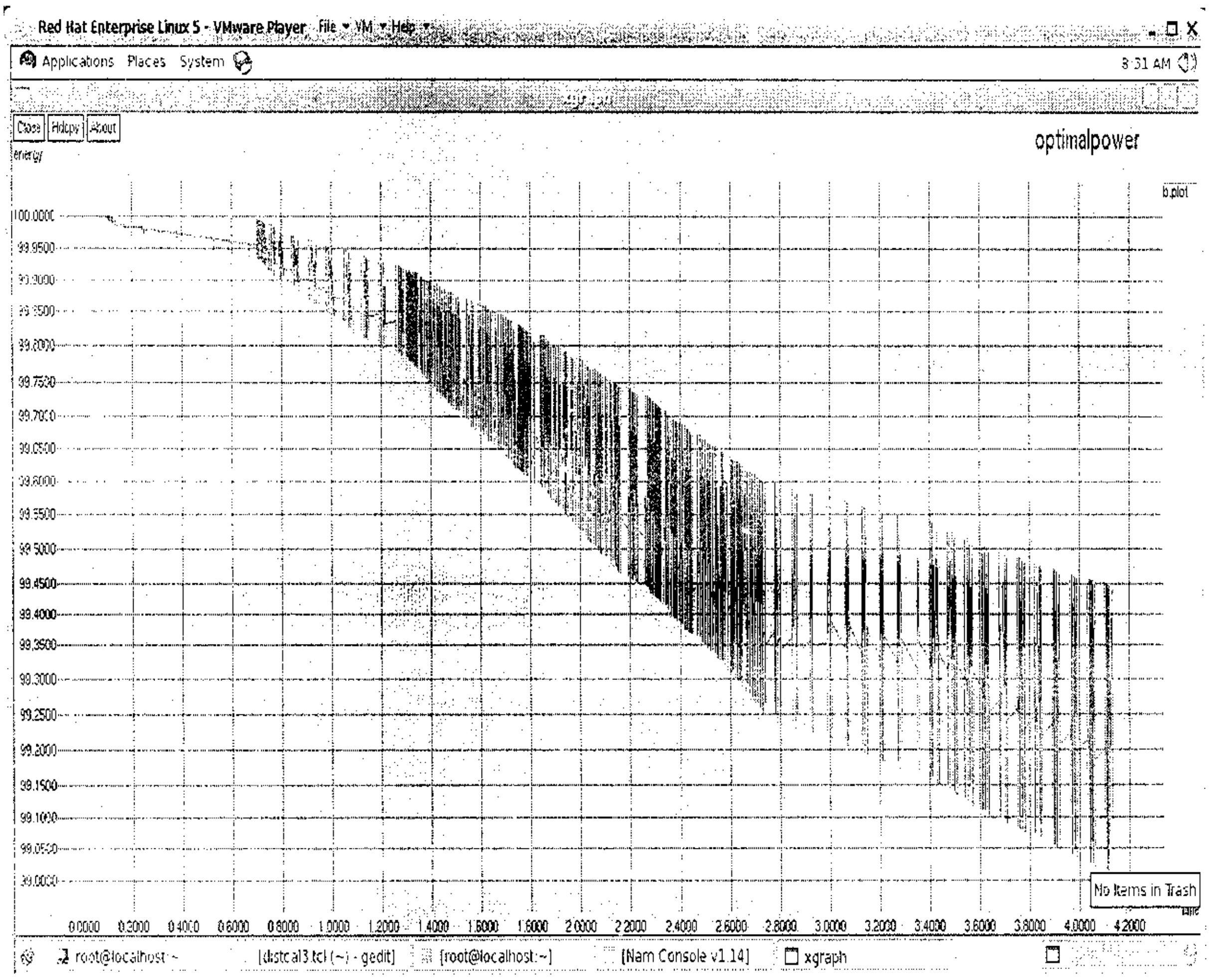


Datatransfer at sensornode at 8 is failed

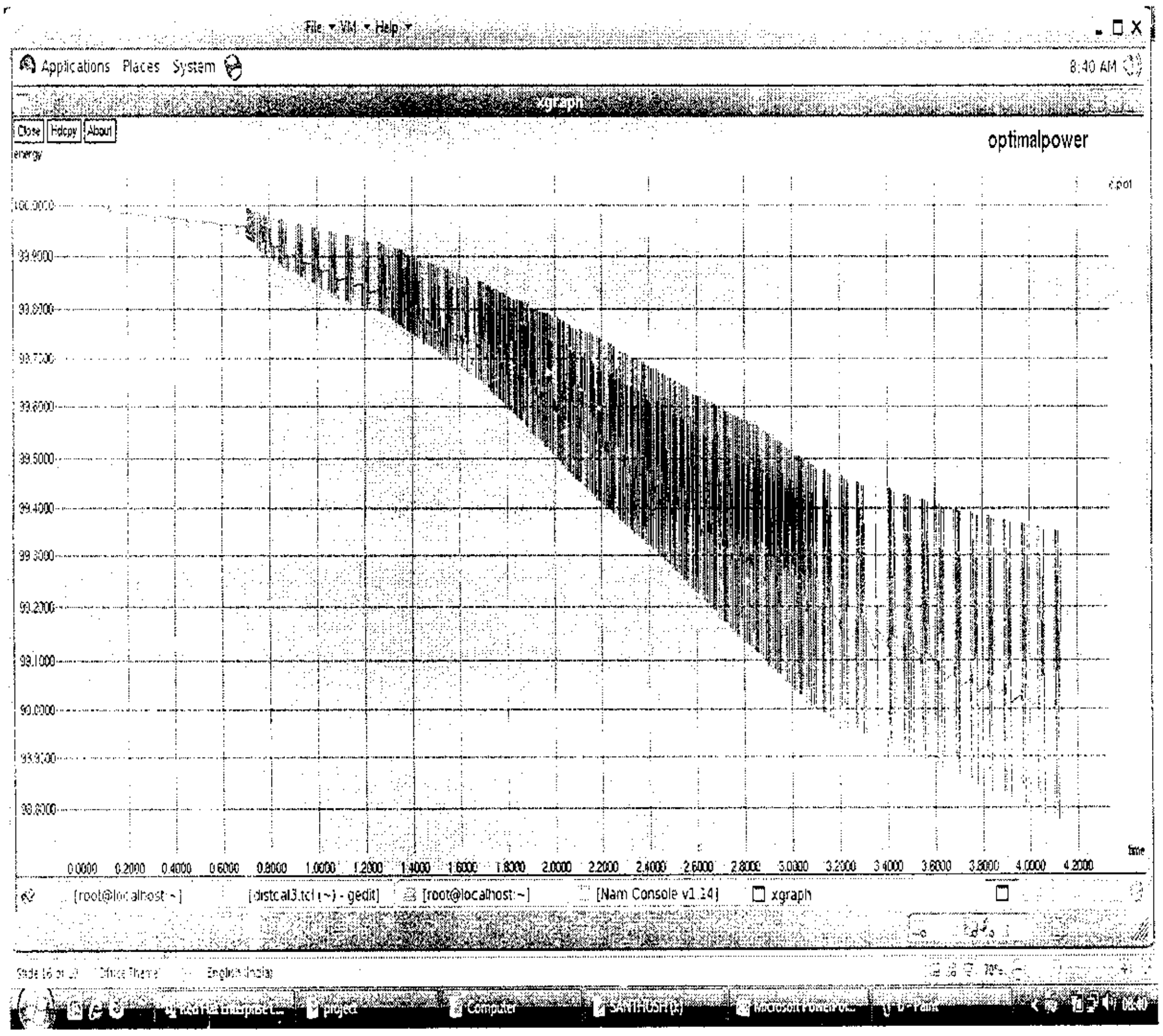
Conneted to clusterhead 1

ENERGY USAGE AFTER NODE FAILURE

BY SELECTIVE REASSIGNMENT



BY COLLECTIVE REASSIGNMENT



REFERENCES

[1] *IEEE 802.15 Wpan Task Group 4 (tg4)*, [Online]. Available:www.ieee802.org/15/pub/TG4.html

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp.102–114, Aug. 2002.

[3] S. Bandyopadhyay and E. J. Coyle, "Minimizing communication costs in hierarchically-clustered networks of wireless sensors," *J. Comput. Netw.*, vol. 44, no. 1, pp. 1–16, Jan. 2004.

[4] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[5] C. F. Chiasserini, I. Chlamtac, P. Monti, and A. Nucci, "An energy efficient method for nodes assignment in cluster-based ad hoc networks," *J. Wireless Netw.*, vol. 10, no. 3, pp. 223–231, May 2004.

[6] T. Moscibroda and R. Wattenhofer, "Maximizing the lifetime of dominating sets," presented at the IEEE WMAN, Denver, CO, 2005.

[7] A.D.Amisand R. Prakash, "Load balancing clusters in wireless adhoc networks," in Proc.3rdIEEESymp.Application-Speci?cSyst.Software Eng.Technol.,Richardson,TX,2000,pp.25–32.

[8]A.D.Amis,R.Prakash,T.H.P.Vuong,andD.T.Huynh,"Max-min d-clusterformationinwirelessadhocnetworks,"in Proc.IEEE INFOCOM, TelAviv,Israel,2000,pp.32–41.

[9]S.BandyopadhyayandE.J.Coyle,"Anenergyef?cienthierarchical clustering algorithm for wireless sensor networks,"in Proc.IEEEINFOCOM,SanFrancisco,CA,2003,vol.3,pp.1713–1723.

[10]S.BandyopadhyayandE.J.Coyle,"Minimizingcommunicationcosts inhierarchically-clusterednetworksofwirelesssensors," J.ComputNetw.,vol.44,no.1,pp.1–16,Jan.2004.