ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLOGY

PERFORMANCE EVALUATION OF WIRELESS SENSOR NETWORK ROUTING PROTOCOLS FOR CRITICAL CONDITION MONITORING APPLICATIONS

A thesis submitted to the School of Graduate Studies of Addis Ababa University in partial fulfillment of the Degree of Masters of Science in Computer Engineering

By

GEZAHEGN GELETA

Advisor

Dr. MANOJ V.N.V

ADDIS ABABA

October, 2007

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLOGY

PERFORMANCE EVALUATION OF WIRELESS SENSOR NETWORK ROUTING PROTOCOLS FOR CRITICAL CONDITION MONITORING APPLICATIONS

BY

GEZAHEGN GELETA

Approval by Board of Examiners

Dr. Mengesha Mamo	_____
Chairman, department of graduate committee	Signature
Dr. Manoj V.N.V	_____
Advisor	Signature
Dr. Hailu Ayele	_____
Interna Examiner	Signature
Dr. Fisha Mekuriya	_____
External Examiner	Signature

Declaration

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

Name : _____Gezahegn Geleta___

Signature: _____

Place : _____Addis ababa_____

Date : _____09/11/2007_____

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

Name : ____Dr. Manoj V.N.V____

Signature: _____

Place : _____Addis ababa_____

Date : _____09/11/2007_____

Acknowledgement

First of all, I would like to thank my advisor Dr. Manoj V.N.V for his guidance and advice until the end of this work. Throughout, he provided me all the supports I needed for this thesis work. This thesis work wouldn't have been possible without the endless supports and creative ideas from him.

I would also like to thank all the electrical and computer engineering department laboratory technicians, who were all generous and cooperative.

Last but not least I would like to offer my thanks to all the friends here who have had good and constructive suggestions for my work.

Contents Page

Chapters:

List of figures

Figure **Page**

List of Tables

Table **Page**

List of Acronyms

ACA	Autonomous Component Architecture
ACQUIRE	ACtive QUery forwarding in sensor Networks
ADC	Analog to Digital Converter
ADE	Average Dissipated Energy
AEED	Average End to End Delay
AODV	Adhoc On demand Distance Vector
APDR	Average Packet Delivery Ratio
APS	Ad hoc Positioning System
ARP	Address Resolution Protocol
BS	Base Station
CADR	Constrained Anisotropic Diffusion routing
CPU	Central Processing Unit
DD	Directed Diffusion
DE	Dissipated Energy
DSP	Digital Signal Processing
EAR	Energy Aware Routing
EED	End to End Delay
EEPROM	Electrically Erasable Programmable Read Only Memory
EPROM	Erasable Programmable Read Only Memory
FCFS	First Come First Served
GAF	Geographic Adaptive Fidelity
GEAR	Geographic and Energy Aware routing
GOAFR	Greedy Other Adaptive Face Routing
GPS	Global Positioning System
HPAR	Hierarchical Power Aware Routing
HPEQ	Hierarchical Periodic, Event driven and Query based
INET	Inter NETworking
ISM	Industrial scientific and medical
Jacl	JAva tCL
LEACH	Low-Energy Adaptive Clustering Hierarchy
LL	Link Layer

MAC	Medium Access Control
MANET	Mobile Ad hoc NETwork
MECN	Minimum Energy Communication Network
MEMS	Micro Electro-Mechanical Systems
MMSPEED	Multi path and Multi SPEED
NES-C	Nested C
PDA	Personal Digital Assistances
PDR	Packet Delivery Ratio
PEGASIS	Power Efficient GAthering in Sensor Information Systems
PEQ	Periodic Event driven and Query based
QoS	Quality of Service
RERR	Route ERRor
RF	Radio Frequency
RP	Routing protocol
RREP	Route REPly
RREQ	Route REQuest
SAR	Sequential Assignment Routing
SPIN	Sensor Protocols for Information via Negotiation
SRAM	Static Random Access Memory
TCL	Tool Code Language
TDMA	Time Division Multiple Access
TEEN	Threshold sensitive energy efficient sensor network protocol
Tiny-OS	Tiny Operating System
TTL	Time To Live
UART	Universal Asynchronous Receive/Transmit
WSN	Wireless Sensor Network

Abstract

Sensor networks are increasingly being deployed, either randomly or manually, for monitoring of physical environments subjected to different applications such as medical, military, agriculture, industry, transport etc. One of the most important applications of a wireless sensor network is critical conditions monitoring. In cases like critical condition monitoring application, it is important that information can be sensed from the physical environment, where the sensor network is deployed, while the emergency state is in progress.

For critical conditions such as fire, leaking of toxic gases and explosions, the system must be fast enough to respond for foreseen events (targets) with in a fraction of seconds. A great challenge to these networks is to provide a fast, reliable, and fault tolerant channel for events diffusion even in the presence of emergency conditions that can lead to node failures and path disruption, like dust, fire etc, to the sink that receives those events.

The main aim of this thesis work is to discuss in detail and evaluate the performance of different routing protocols selected from wireless sensor network (WSN) and mobile ad hoc network (MANET) for critical conditions monitoring applications with the help of important metrics.

The result shows that multi-path and multi-speed (MMSPEED) routing protocol has a very good performance of events delivery with in a short end-to-end delay and high ratio of packet delivery by creating a reliable path for packets reaching to the destination in case of critical conditions monitoring applications with the limitation of moderate power dissipation when compared to other routing protocols.

CHAPTER 1: INTRODUCTION

Sensors which are part of the transducers, measure ambient conditions in the environment that surround them and then transform these measurements into signals that can be processed to reveal some characteristics about phenomena located in the area around these sensors. A large number of these heterogeneous or homogenous sensors can be networked in many applications that require unattended operations, hence producing a WSN.

WSNs contain hundreds or thousands of sensor nodes, and these sensors have the ability to communicate either among one another or directly to an external base station (BS). A greater number of sensors allows for sensing over larger geographical regions with greater accuracy. However, as the network size increases, then the coverage also increases, transmitting the data to the BS become impossible because of the limitation of radio range of the nodes so that the communication will have to be conducted through intermediate nodes, multi-hop communication (I.F. Akyildiz et al., 47).

There are different factors that affect WSNs. One of the major factors is their deployment (Krishnamurthy et al., 43). As soon as the deployment is established, the sensors configure themselves to form a network and start their work. Deployment of a sensor network in different applications can be in random fashion (e.g., dropped from an airplane in a disaster management application) or manual (e.g., fire alarm sensors in a facility or sensors planted underground for precision agriculture).

The main limitations of sensor nodes are constraints in energy supply, low computation power and bandwidth. Such constraints combined with a typical deployment of large number of sensor nodes have posed many challenges to the design, manufacturing and management of sensor networks and also routing becomes very difficult since there is no fixed infrastructure between them to communicate. With a lot of constraints in WSNs, a number of protocols have been proposed to minimize the above limitations.

1.1 Background

Wireless communication is one of the latest technologies that play a great role in modern life, from global cellular telephone systems to local and even body or personal-area networks (J. Stankovic et al., 49). Recently invented wireless devices have considerably increased the need for networking and provisioning intercommunication between these devices for various applications.

Broadly, wireless networks can be classified into two categories. The cellular and wireless local area networks can be termed as infrastructure-based wireless networks, as they rely on beforehand deployed infrastructures and the second category, termed infrastructureless or rapidly-deployable wireless networks that do not need any infrastructure support for deployment. Rapidly-deployable networks include MANETs and WSNs.

Both categories, infrastructure-based and infrastructure-less, have their own pros and cons.

- Infrastructure based networks carry data and voice from source to destination with good quality of service but they require infrastructure
- Infrastructureless networks even though they are constrained with limitation in range, bandwidth and power, have a number of advantages.
  - MANETs have used in:-
    - ✓ Battlefields
    - ✓ Disaster recovery areas
    - ✓ Intercommunication between islands or in difficult terrains, and
    - ✓ Conferencing with any infrastructure support.
  - WSNs have a wide range of applications in but not limited to:-
    - ✓ Monitoring environments
    - ✓ Sensitive installations, and
    - ✓ Remote data collection and analysis.

In both MANET and WSN the nodes act both as hosts as well as routers. They operate in a self-organizing and adapting manner.

1.2 Problem description

WSNs are increasingly being deployed for accurate monitoring of physical environments subjected to critical conditions such as fire, leaking of toxic gases in an industrial plant and explosions at mining area etc. The order in which events are received from the sensing environment can determine the correct interpretation of what is occurring in the physical environment being monitored. Therefore, the ordering of events in WSNs can be a major requirement for critical monitoring applications to avoid ambiguities, increasing monitoring accuracy.

A great challenge to these networks, WSNs, is to provide a fast, reliable and fault tolerant channel for the diffusion of events even in the presence of emergency conditions that can lead to node failures, like dust and fire, and path disruption to the sink that receives those events.

In cases like critical condition monitoring, it is important that information can be "sensed' from the physical environment while the emergency state is in progress. However, in order to keep the information flowing from the sensors during the emergency, a WSN solution has to cope with the failure of sensor nodes (sensors can be burnt, have their propagation interrupted by interferences, such as water or dense smoke present in the environment, can be malfunctioning, exhausted their powers etc). Thus, WSN solutions for such environments have to be fault tolerant and reliable, and to provide low latency, besides fast reconfiguration during path breakage and use energy saving mechanisms. For most of these activities, the routing protocol plays a great role.

Route information in WSNs is very much unstable and hence makes routing complicated due to lack of a common access point for nodes, i.e. infrastructure-less , limited power and bandwidth, limited computing power of central processing unit (CPU) so that the routing protocols should minimize the usage of these valuable resources. In order to address the above problems a lot of routing protocols have been introduced, all of which typically perform well in some situations while having significant weakness in other cases. Therefore, performance evaluation of various existing routing protocols is crucial to enhance the existing one or to design an efficient new routing protocol for critical condition monitoring application and also help to standardize the

existing ones. Therefore the main aim of this thesis work is to study and evaluate the selected routing protocols for critical condition monitoring applications based on performance metrics: average delay, average packet delivery ratio and average dissipated energy which are directly affected by this application.

1.3 Related work

A number of routing protocols are proposed for WSNs. And it is known that all the proposed protocols are not efficient and do not qualify all the desired property for critical conditions monitoring applications.

Despite of the multiplicity of protocols, few comparisons between the different protocols have been made. In this section, related works on comparison of routing protocols for either real time or critical condition monitoring applications will be presented.

- The authors (Boukerche et al., 7) have compared their new protocol, hierarchical, periodic, event driven and query based (HPEQ), with periodic, event driven and query based (PEQ) using some metrics for critical condition monitoring applications. The major problem with their works was, they have not taken into consideration many other protocols that are used for real time application. And also they did not consider existing MANET protocols into account.

- Here also after implementing SPEED routing protocol (T. He et al., 2) for soft real-time monitoring application, the authors compare it with its former versions and with some MANET routing protocols. But they did not consider some metrics such as average dissipated energy, average packet delivery ratio which were helpful for critical condition monitoring applications.

- Here also the authors implement MMSPEED routing protocol (Emad Felemban et al., 13) and compare it with SPEED routing protocol. But again in here the main intension was to evaluate their new algorithms for reliability and timeliness so that the metrics they selected were not totally related for critical conditions.

- In (Nam N. et al., 23), a comparison of four WSN protocols: MultiHopRouter, TinyAODV (Ad-hoc On-demand Distance-Vector), GF (Greedy Forwarding), and GF-RSSI (Greedy Forward with Received Signal Strength Indication) was done. In this work the performance measurement was conducted on a WSN testbed to select the best protocol for medical application research.

All these comparisons except the last were done to check the performance of the author's new designed protocol with the one they expected to be designed formerly for similar applications. However, they are still not quite sufficient to make decisions as to which routing protocol is better for critical condition monitoring application.

1.4 Objectives

General Objective

The general objective of this thesis work is to study, evaluate and select a routing protocol for wireless sensor network from the existing one, which can have a better performance for critical condition application through thorough evaluation and testing of the protocols for different scenarios.

Specific Objectives

The specific objectives of the thesis includes but not limited to:-

- Designing and developing simulation model
- Conduct a simulation with different metrics and network scenario
- Analyzing simulation results for the required application.
- Performing performance evaluation and deriving a conclusion.

1.5 Methodology

To accomplish the objectives of this thesis work, the following steps were followed:

- **Literature review**

  At this step, assessing any published work or survey a literature to gather any information about the study area relevant to the research work has done.

- **Study MANETs and WSNs**

  This step will give an understanding of the emerging new technologies in general from the wireless communication point of view. This is the background information required to understand the subject matter of this thesis work. In order to understand about WSNs, it is a simple practice starting from MANETs which are the base for WSNs.

- **Study routing protocols for MANETs and WSNs**

  This step allows an understanding of how the routing protocols for both networks work and what are the main characteristics and differences. This includes how:-

  - the routing table will be formed
  - path will be selected
  - route request/reply will be done, etc

- **Study and modify the JSIM simulator**

  JSIM is a scalable, discrete and component based simulator used for wireless networking in an ad-hoc manner. To design, analysis and compare the results for this thesis work, different routing protocols should be implemented on this simulator and the simulator should also be modified to handle different routing protocols with their specifications. The simulator supports only ad-hoc on demand distance vector (AODV) protocol.

- **Simulation/ Performance evaluation**

  Once a detailed discussion of MANET/WSN routing protocols is presented and necessary modification and implementations have been done, the next step is preparing a model for each routing protocols and analyze the effect for critical condition monitoring application with the help of different metrics for the given application. These are: *Average packet-delivery-ratio, Average end-to-end delay* and *Average dissipated-energy.*

- **Result Analysis**

  At this level, the outputs obtained from the simulations for the selected protocols for different metrics and scenarios will be analyzed up to the limit of the simulator scalability.

1.6 Outline of the thesis

The thesis work comprises of seven chapters. The next chapter covers the study of a WSN, the architecture, components and applications. It also covers the comparison between WSN and MANET and two examples of commercially available sensor systems.

Chapter three deals with the main design issues and study routing protocols of MANET and WSNs. This chapter covers different types of protocols including their architecture, classification and detail explanation of the proposed protocols for this work.

Chapter four discusses the simulator for this work. In this chapter the main components of the simulator i.e. sensor node, sink node and target node will be discussed specifically. And the main classes of the simulator will also be discussed.

Chapter five discusses the modifications that have been done on the simulator to suit for the required thesis work. In this chapter discussion was focused only on the main modification issues like implementing routing protocols, medium access control (MAC) layer, power model etc.

Chapter six discusses about testing and the testing criteria and also discusses the results of the simulation based on the metrics which were selected for this application.

Finally, Chapter seven is concerned with conclusion of what is done in this thesis and recommendation of what more can be done for future related with this area that is not covered in this thesis.

CHAPTER 2: WIRELESS SENSOR NETWORKS

2.1 INTRODUCTION

A WSN, a distributed real-time system, is a collection of nodes organized into a cooperative network intended to perform a specific task. Each node consists of processing capability (one or more microcontrollers, CPUs or digital signal processing (DSP) chips), multiple types of memory (program, data and flash memories), radio frequency (RF) transceiver (usually with a single omni-directional antenna), power source (e.g., batteries and solar cells), and accommodate various sensors and actuators. The nodes communicate wirelessly and often self-organize after being deployed in an ad hoc fashion. Systems of 1000s or even 10,000 nodes are anticipated in a single network (f. l. lewis, 15), [47].

When we compare it with wired distributed systems, the latter have unlimited power, are not real-time, have user interfaces such as screens and mice, have a fixed set of resources, treat each node in the system as very important and are location independent. In contrast, for WSNs, the systems are wireless, have scarce power, are real-time, utilize sensors and actuators as interfaces, have dynamically changing sets of resources, aggregate behavior is important and location is critical. Many WSNs also utilize minimal capacity devices which places a further strain on the ability to use past solutions.

Currently, WSNs are beginning to be deployed at an accelerated rate for different applications to ease life of a human being. It is not unreasonable to expect that in the near future the world will be covered with WSNs with access to them via the Internet. This new technology is exciting with unlimited potential for numerous application areas including environmental, medical, military, transportation, entertainment, crisis management, homeland defense, and smart spaces.

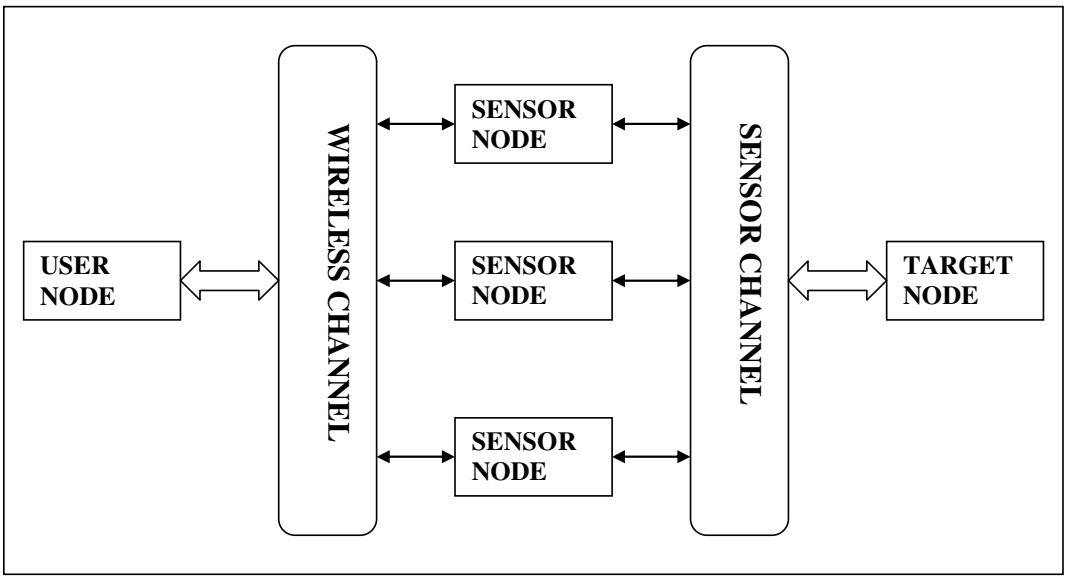


Fig 2.1 Block diagram of WSN [15]

2.2 **Transducers and Physical Transduction Principles** (Olivier Bezet et al., 6)**,** [15], (Jason Lester Hill, 22), (A. Ananda et al., 26)

A transducer is a device that converts energy from one domain to another. In our application, it converts the quantity to be sensed from the environment into a useful signal that can be directly measured and processed which are generally voltages or currents. A transducer can be either a sensor or an actuator, where a sensor is a device that converts a non electrical input quantity into an electrical output signal and an actuator is a device that converts an electrical signal into non electrical quantity.

The three main types of sensory transduction principles are:

2.2.1 **Micro electro-mechanical Systems (MEMS) sensors:** These are sensors which use the effects of either ***piezoresistive*** or ***piezoelectric***t. Such sensors are by now very well developed and are available for most sensing applications in wireless networks. Mechanical Sensors include those that rely on direct physical contact.

2.2.2 **Magnetic and Electromagnetic Sensors:** are sensors which use either ***Hall*** or ***magnetoresistive*** effects do not require direct physical contact and are useful for detecting proximity effects.

2.2.3 **Thermo-Mechanical Transduction sensors:** which uses the principles of either ***thermoresistive*** or ***thermoelectric*** effect is used for temperature sensing and regulation in homes and automobiles.

2.3 Classification of sensors [6], [15], [22], [26]

Based on the above principles, sensors can be categorized into:

- **Magnetic Sensors** can be used to detect the remote presence of metallic objects.
- **Thermal Sensors** are a family of sensors used to measure temperature, heat flow or heat flux.
- **Optical Transducers** convert light to various quantities that can be detected. Solar cells are large photodiodes that generate voltage from light.
- **Chemical and Biological Transducers** cover a very wide range of sensors that interact with solids, liquids, and gases of all types and uses for environmental monitoring, biochemical warfare monitoring, security area surveillance, medical diagnostics, implantable biosensors, and food monitoring.
- **Electrochemical Transducers** rely on currents induced by oxidation or reduction of a chemical species at an electrode surface. These are among the simplest and most useful of chemical sensors.

2.4. Node hardware components [15], [22], [26], (J. Heidemann et al., 34)

The implementation and development of the wireless sensor node must consider critical criteria such as design, according to a very well studied and analyzed application. This is required to specify and define the profile of the elements (hardware) as well as the characteristics and

methods (software and programming model). This offers the best flexibility and efficiency during the operation. Besides, it has to be taken into consideration that the sensor nodes in a WSN must be small in size, cheap, energy efficient, equipped with sensor(s), good computation performance, adequate size-storage and suitable communication facilities.

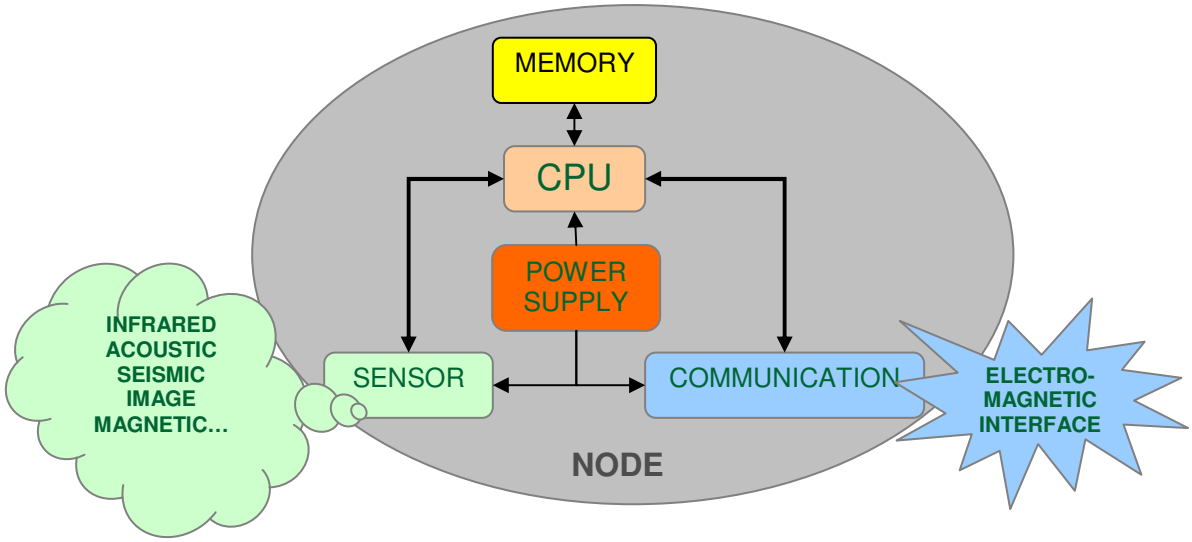


Fig 2.2 Sensor node internals

During the selection of the hardware elements for a wireless sensor node, the final application of the network defines the size, cost, and energy requirements. The basic parts of the hardware of the sensor nodes are as follows:-

2.4.1. **Processor:-**is in charge of processing all relevant data and executing the code that describes the behavior of the sensor node. Some common tasks of the processor are acquisition, preprocessing and processing of the incoming and outgoing information. A further task is the implementation of the routing of information according to different conditions of performance and reliability. Microcontrollers (consisting of a processing unit, memory, and peripheral modules like analog-digital converter (ADC), universal asynchronous receive/transmit (UART), timers, interferences connected via bus) or DSPs are usually exploited to perform this role.

2.4.2. **Memory: -** is in charge to store programs and data. Commonly random access memory (RAM) is used to store intermediate and temporary samples or packets from other nodes. Memories like electrically erasable programmable read only memory (EEPROM) or flash are used to store the program code. The energy consumption by the memory is an important factor for the appropriate design of the wireless sensor node.

2.4.3. **Sensors: -** are the interfaces to the physical world which senses the environment and convert them to a fundamental data (mostly voltage or current) for further processing within the processor. There are a great variety of sensors that are available and in use with respect to the sensing radiation directions; but the choice have to be made considering the size and energy consumption that is required for its operation. Common categorizations of the sensor are:

- **Passive, omni directional sensors** can measure a physical quantity at the point of the sensor node without actually manipulating the environment. Examples are *thermometers, light sensors, accelerometers, microphones, humidity* and *pressure sensors*.
- **Passive, narrow-beam sensors** have a well-defined notion of direction for measurement. These sensors demand a defined orientation in order to measure the medium. Examples are *ultrasonic sensors* or *cameras*.
- **Active sensors** are continuously taking samples of the environment. Examples are *sonar, radar* and *seismic sensors*.

2.4.4 **Communication elements** are the hardware to enable the networking capability of the sensor nodes. Some usual options for communication between this hardware and the environment are radio frequencies, optical communication, etc. RF is generally used because it provides a long range of transmission and reception at high data rates with acceptable error rates for the required energy. One important feature is that it does not require a direct line of sight between two neighbors.

2.4.5 **Power supply:** there exists a large quantity of power supply options for a sensor node. The most common option is the use of batteries; other options are scavenging energy from the environment where the sensor node is exposed, the most popular example is the solar cells.

2.5 **Comparison of MANETs with WSNs** (C. E. Perkins, 5),[15],[22],[47]

As mentioned earlier wireless communication in general is classified into two main categories, infrastructure based and infrastructureless, and again infrastructureless wireless networks are also divided into two groups called WSNs and MANETs. Both are equivalent networks built for different purposes. WSNs and MANETs are self-configuring networks where nodes, connected by wireless links, can move freely and thus the topology of the network changes constantly. Both have similarities and differences.

The ***similarities*** are: -

- Both are *distributed infrastructure-less wireless networks*
- Routing between two nodes may involve the *use of intermediate relay nodes,* called *multihop routing*.
- Both are usually battery-powered and therefore there is a big concern on *minimizing power consumption*.
- Both networks use a wireless channel placed in an unlicensed spectrum that is *prone to interference* by other radio technologies operating in the same frequency.
- *Self-management* is necessary because of the distributed nature of both networks.

The ***differences*** are (Borrong Chen et al., 28): -

- MANET nodes are always in touch *by human beings* (e.g., laptop computers, PDAs, mobile radio terminals, etc.) whereas sensor networks do not focus on human interaction but instead focus on *interaction with the environment*.
- Sensor network nodes are usually embedded in the environment to *sense some phenomenon* and possibly actuate upon it whereas nodes in MANETs are used for *data and information exchange*.
- The *number of nodes* in sensor networks, as well as the density of deployment, can be orders of magnitude higher than in MANETs.
- Due to their application nature, e.g. on top of a volcano or fire, nodes in WSNs may fail frequently. In this case *reconfiguration mechanisms* will have to be used, so the network design should consider that nodes are prone to failure.

- In rare cases nodes in WSNs are mobile, but in the *majority of applications nodes remain static*, so some issues that are important in MANETs may not be of great importance in WSNs.
- For some applications, instead of the ID (e.g., the address) of individual nodes location becomes a more important attribute. In general, communication paradigms are affected by the application-specific nature of sensor networks.
- WSNs are mostly deployed and *owned by a single user* (at BS) whereas in MANETs many users can participate at a time.
- Nodes for MANETs are *interfaced with or near to the human beings* so that no question of energy and bandwidth conservation arises but this is a primary concern in WSNs because for the latter application nodes deploy far from human beings.
- In WSNs *data comes from multiple nodes to the BS* so that congestion and maximum energy usage may occur at/around the BS. Some techniques or methods should be employed to avoid this problem.

2.6 **Commercially Available Wireless Sensor Systems** [15]

Many commercially available wireless communication nodes as a full set and ready for use are available from different vendors with different technologies. Two of them are described below.

2.6.1 **Crossbow Berkeley Motes** [21] are the most versatile WSN devices on the market for prototyping purposes. Nodes come with five sensors installed: *temperature, light, acoustic (Microphone), acceleration/Seismic and magnetic.* These are especially suitable for surveillance networks for personnel and vehicles tracking. Different sensors can be installed other than the above mentioned if desired. Low power and small physical size enable placement virtually anywhere.

Since all sensor nodes in a network can act as BSs, the network can self configure and has multi-hop routing capabilities. The operating frequency is industrial, scientific and medical (ISM) band of 902-930MHz (center frequency of 916 MHz), with a data rate of 40 Kbits/sec. and a range of 30 ft to 100 ft coverage for RF signal. Each node has a low power microcontroller processor with a speed of 4MHz, a flash memory with 128

Kbytes, and static RAM (SRAM) and EEPROM of 4K bytes each. The operating system is Tiny-OS; a tiny micro-threading distributed operating system developed by UC Berkeley and used as an operating system for most WSNs, with a NES-C (Nested C) source code language (similar to C). Installation of these devices requires a great deal of programming.

2.6.2 **Microstrain's X-Link Measurement System** [24] are the easiest system to get up and running and to program. The frequency used in here also is the ISM band 916 MHz, which lies in the US license-free band. The sensor nodes are multi-channel, with a maximum of 8 sensors supported by a single wireless node at a time. There are three types of sensor nodes: s-*link (strain gauge), g-link (accelerometer) and v-link (supports any sensors generating voltage differences).*

The sensor nodes have a pre-programmed erasable programmable read only memory (EPROM), so a great deal of programming by the user is not needed. Onboard data storage is 2MB. Sensor nodes use a 3.6-volt lithium ion internal battery (9V rechargeable external battery is supported). Since a single receiver (BS) has a unique 16-bit address, a maximum of 2^{16} nodes can be addressed from it. The RF link between BS and nodes is bi-directional and can cover 30 meter range with a 19200 baud rate.

2.7 **Applications of WSNs** [15], (Victor Shnayder et al., 25), (C. Shen et al., 35)

WSNs have so many applications when compared with other networks because of their tiny size and deploy them from any where and self-configuring nature. The main application classification can be:

- ***Military Applications***
  - monitoring friendly forces
  - monitoring equipment
  - battlefield surveillance
  - reconnaissance of opposing forces and terrain

- ***Environmental Monitoring***
  - flood/forest fire detection
  - space exploration
  - biological attack detection
  - soil condition monitoring
  - seismic activity detection
- ***Commercial Applications***
  - home/office smart environments
  - health applications
  - environmental control in buildings
  - monitoring an industrial plant
  - inventory control
  - weather monitoring
- ***Tracking***
  - targeting in intelligent ammunition
  - tracking of doctors and patients inside a hospital
- ***Detecting ambient conditions***
  - temperature
  - movement
  - Sound
  - Light
  - presence of certain objects
  - intrusion detection

CHAPTER 3: ROUTING PROTOCOLS IN WSNs AND MANETs

3.1 INTRODUCTION

Routing is a technique used to transmit data from source to the destination through direct or single/multihop nodes. Since a distributed network like WSN has multiple nodes and services many messages, and each node is a shared resource, many decisions must be made because of multiple paths from the source to the destination in order to select the most reliable path for the message to reach the destination. Therefore, message routing is an important topic for WSNs.

Routing in WSNs is very challenging due to the inherent characteristics that distinguish these networks from other wireless networks like MANETs or cellular networks (Abd-El-Barr et al., 10), (Jamal N. Al-karaki et al., 16), (Holger Karl et al., 29): -

- Traditional internet protocol (IP)-based protocols may not be applied due to the relatively large number of sensor nodes.
- Sometimes getting the data is more important than knowing the IDs of which nodes sent the data.
- Almost in all applications the flow of sensed data is from multiple sources to a particular BS.
- Careful resource management is required since sensor nodes are tightly constrained in terms of energy, processing, and storage capacities.
- In most application scenarios, nodes are generally stationary after deployment except for maybe a few mobile nodes.
- Sensor networks are application-specific at design time.
- Since data collection is normally based on the location, position awareness of sensor nodes is important.
- Data collected by many sensors in WSNs is typically based on common phenomena, so there is a high probability that this data has some redundancy.

Due to such differences, the routing protocols should handle such characteristics. Many new algorithms have been proposed for the routing problem in WSNs. These routing mechanisms have

taken into consideration the inherent features of WSNs along with the application and architecture requirements.

3.2 **Routing challenges and design issues in WSNs**

Despite the innumerable applications of WSNs, these networks have several restrictions, such as limited energy supply, limited computing power, and limited bandwidth of the wireless links connecting sensor nodes so that we have to consider these limitations at design time (Chi-Fu Huang et al., 38). One of the main design goals of WSNs is to carry out data communication while trying to prolong the lifetime of the network and prevent connectivity degradation by employing aggressive energy management techniques.

The design of routing protocols in WSNs is influenced by many challenging factors [16]. These factors must be overcome before efficient communication can be achieved in WSNs. Some of the routing challenges and design issues that affect the routing process in WSNs are (qinglan li et al., 37), (Jang-Ping Sheu et al., 41):

3.2.1 **Node deployment:** In WSN node deployment can be either manual (uniformly deployed) or randomized based on the required applications. In manual deployment, the sensors are manually placed and data is routed through predetermined paths. However, in random node deployment, the sensor nodes are scattered randomly, for example scattering in a disaster area from the aircraft, creating an ad hoc routing infrastructure.

3.2.2 **Energy consumption:** One of the main limiting factors for WSNs is their limited power supply. Sensor nodes can use up their limited supply of energy performing computations and transmitting information in a wireless environment at sending or routing of data. As such, energy-conserving forms of communication and computations are essential to overcome such limitations. Sensor node's lifetime is strongly dependence on battery life time.

3.2.3 **Data reporting method:** Data reporting in WSNs is application-dependent and also depends on the time criticality of the data. Data reporting can be categorized as either time-driven (for

applications that require periodic data monitoring), event driven (for applications that react immediately to sudden and drastic changes), query-driven, or a hybrid of all these methods.

3.2.4 **Node/link heterogeneity:** In here also based on the application of the network, the nodes can be homogeneous (i.e., have equal capacities in terms of computation, communication, and power) for a specific application or it can be heterogeneous set of sensors for example a diverse mixture of sensors for monitoring temperature, pressure, and humidity of the surrounding environment, detecting motion via acoustic signatures, and capturing images or video tracking of moving objects etc which brings about different computing power, power consumption for processing and communication, data reading and reporting at different rates.

3.2.5 **Fault tolerance:** Some of the reasons for the nodes to fail are due to lack of power, physical damage, or environmental interference. The failure of sensor nodes with in the network should not affect the overall task of the sensor network. If many nodes fail, medium access control and routing protocols must accommodate formation of new links and routes data to the BSs.

3.2.6 **Scalability:** The number of sensor nodes deployed in the sensing area may be on the order of hundreds or thousands, or more as per required by the application and coverage area. Any routing scheme must be able to work with this huge number of sensor nodes. In addition, sensor network routing protocols should be scalable enough to respond to events in the environment.

3.2.7 **Network dynamics:** Even though in many applications sensor nodes are assumed fixed, in some applications either or both the BS and/or sensor nodes can be mobile and even the observed or sensed phenomenon can be mobile (e.g., target detection/ tracking application). As such, routing messages from or to moving nodes is more challenging since route and topology stability become important issues, in addition to energy, bandwidth, and so forth. On the other hand, sensing fixed events allows the network to work in a reactive mode (i.e., generating traffic when reporting), while dynamic events in most applications require periodic reporting to the BS.

3.2.8 **Transmission media:** In a multihop sensor network, communicating nodes are linked by a wireless medium. Thus the traditional problems associated with a wireless channel (e.g., fading, high

error rate) may also affect the operation of the sensor network. In general, the required bandwidth of sensor data will be low, on the order of 1–100 kb/s.

3.2.9 **Connectivity:** Sensor nodes are expected to be highly connected even on random distribution. This, however, may not prevent the network topology from being variable and the network size from shrinking due to sensor node failures.

3.2.10 **Coverage:** A given sensor's view of the environment is limited in both range and accuracy; it can only cover a limited physical area of the environment. Hence, area coverage is also an important design parameter in WSNs.

3.2.11 **Data aggregation:** Data aggregation is the combination of redundant data, similar packets from multiple nodes, according to a certain aggregation function (e.g., duplicate suppression, minima, maxima, and average) (J. N. Al-Karaki et al., 45). This technique has been used to achieve energy efficiency and data transfer optimization in a number of routing protocols.

3.2.12 **Quality of service (QoS):** In some applications, like critical or real-time conditions data should be delivered within a certain period of time from the moment it is sensed, or else it will be useless. Therefore, bounded latency for data delivery is another condition for time-constrained applications. However, in many applications, conservation of energy, which is directly related to network lifetime, is considered relatively more important than the quality of data sent. As energy is depleted, the network may be required to reduce the quality of results in order to reduce energy dissipation in the nodes and hence lengthen the total network lifetime. Hence, energy-aware routing protocols are required to capture this requirement (M. Younis et al., 1).

3.3 **Classification of routing protocols** [16], [29], [41]

Since WSNs have many limitations regarding computation power, bandwidth and power consumption, different routing protocols were designed and implemented to compensate these drawbacks.

Routing protocols in WSNs can be divided into two broad categories as based on either *network structure or Protocol operation.*

3.3.1 Network structure

The underlying network structure can play a significant role in the operation of the routing protocol in WSNs. The protocols that fall into this category can also be divided into three sub groups. These are:

3.3.1.1 ***Flat-based routing***: In this group, each node typically plays the same role and sensor nodes collaborate to perform the sensing task by themselves. Because of the large number of such nodes, it is not feasible to assign a global id to each node so that send/receive with this id. This consideration has led to data-centric routing, where the BS sends queries, data with attribute-based naming, to certain regions and waits for data from the sensors located in the selected regions. Examples of routing protocols under this category are:

- *Sensor Protocols for Information via Negotiation (SPIN)*
- *Directed Diffusion (DD)*
- *ACtive QUery forwarding In sensoR nEtworks (ACQUIRE)*
- *Energy-Aware Routing (EAR)*
- *Constrained Anisotropic Diffusion Routing (CADR)*

3.3.1.2 ***Hierarchical-based Routing:*** Hierarchical or cluster based routing methods are well-known techniques which help to get special advantages related to scalability and efficient communication and also utilized to perform energy-efficient routing in WSNs. In a hierarchical architecture, higher-energy nodes(cluster heads) can be selected randomly and used to process and send the information, while low-energy nodes can be used to perform the sensing in the proximity of the target and send to the cluster heads. The creation of clusters and assigning special tasks to cluster heads can greatly contribute to overall system scalability, lifetime, and energy efficiency. Examples of this group are:

- *Low-Energy Adaptive Clustering Hierarchy (LEACH)*
- *Threshold sensitive Energy Efficient sensor Network protocol (TEEN)*
- *Power-Efficient GAthering in Sensor Information Systems (PEGASIS)*
- *Minimum Energy Communication Network (MECN)*
- *Hierarchical Power-Aware Routing (HPAR)*

3.3.1.3 ***Location-Based Routing***: In this group, sensor nodes are addressed by means of their locations, which they got either from GPS or using some other techniques. The distance between neighboring nodes can be estimated on the basis of incoming signal strengths. Relative coordinates of neighboring nodes can be obtained by exchanging such information between neighbors. Examples to this group are:

- *Geographic and Energy-Aware Routing (GEAR)*
- *GEographic DIstance Routing (GEDIR)*
- *Greedy Other Adaptive Face Routing (GOAFR)*
- *Geographic Adaptive Fidelity (GAF)*
- *Sequential Assignment Routing (SAR)*
- *Adhoc positioning system (APS)*

3.3.2 Protocol operation

Different routing protocols have different routing functionality for different applications. Based on their operation or functionality, protocols can be categorized into four classes. These are:

3.3.2.1 ***Multipath Routing Protocols:*** protocols that use multiple paths rather than a single path in order to enhance network performance. These alternate paths are kept alive by sending periodic messages. Hence, network reliability can be increased at the expense of increased overhead in maintaining the alternate paths. But it expenses increased energy consumption due to path setup and traffic generation when data sent from all neighbor nodes. Protocols use multipath routing are:

- *MMSPEED*
- *SPIN*

3.3.2.2 ***Query-Based Routing:*** In this kind of routing, the protocol works using sending/receiving queries. The destination nodes propagate a query for data (sensing task) from a node through the network, and a node with this data sends the data that matches the query back to the node that initiated the query. Usually these queries are described in natural language or high-level query languages. Examples to this class are:

- *SPIN*
- *DD*
- *COUGAR*

3.3.2.3 ***Negotiation-Based Routing:*** These protocols use high-level data descriptors in order to eliminate redundant data transmissions through negotiation. Communication decisions are also made based on the resources available to them. Some examples of the protocols in this group are:

- *SPAN*
- *SAR*
- *DD*

3.3.2.4 ***QoS-based Routing:*** In QoS-based routing protocols, the network have to balance between energy consumption and data quality. In particular, the network has to satisfy certain QoS metrics like delay, energy or bandwidth but not all at a time when delivering data to the BS. To mention some of this group:

- *SAR*
- *SPEED*
- *MMSPEED*

3.4 **Proposed protocols**

From among the existing protocols for WSNs and MANETs, the following are selected for analysis and evaluation (using simulation) of their performance for critical condition monitoring applications.

- *MMSPEED*
- *LEACH*
- *DD*
- *AODV*
- *APS*

The criteria for selecting routing protocols for WSN is based on their design for real time application and AODV is selected from MANETs because of selected as a standard by internet engineering task force (IETF) as a standard and most researches are conducting on it which helps researchers to work on the drawbacks for this application once investigates.

The discussion of each of the above routing protocols in detail about their design issues and implementation techniques are as follows.

3.4.1 **MMSPEED** [13]

3.4.1.1 **Introduction**

One of the protocols in WSNs designed and implemented for QoS application in both reliability and timeliness domain is MMSPEED. To do this, the protocol provides multiple QoS levels in the timeliness domain by guaranteeing multiple packet delivery speed options and in the reliability domain, various reliability requirements are supported by probabilistic multipath forwarding.

3.4.1.2 **Algorithm details**

The main design goals for this routing protocol are:

- *localized packet routing decision* without knowing global network state update or a priori path setup (Lingxuan Hu et al., 4)
- Providing differentiated QoS options in *timeliness and reliability domain*

3.4.1.2.1 **Localized packet routing decisions**

For localized packet routing, each sensor node is assumed to be aware of its geographical location using either global positioning system (GPS) or distributed location services (R. Stoleru et al., 40) and it assumes that the packet destination is specified by a geographic location rather than node ID. The protocol uses the geographic routing mechanism based on location awareness without end-to-end path setup and maintenance and the location information can be exchanged with immediate neighbors with "periodic location update packets." Thus, each node is aware of its immediate neighbors within its radio range and their locations.

3.4.1.2.2 **Timeliness Domain**

For on-time delivery of packets with different end-to-end deadlines from source to destination, the protocol provides multiple delivery speed options that are guaranteed all over the network.

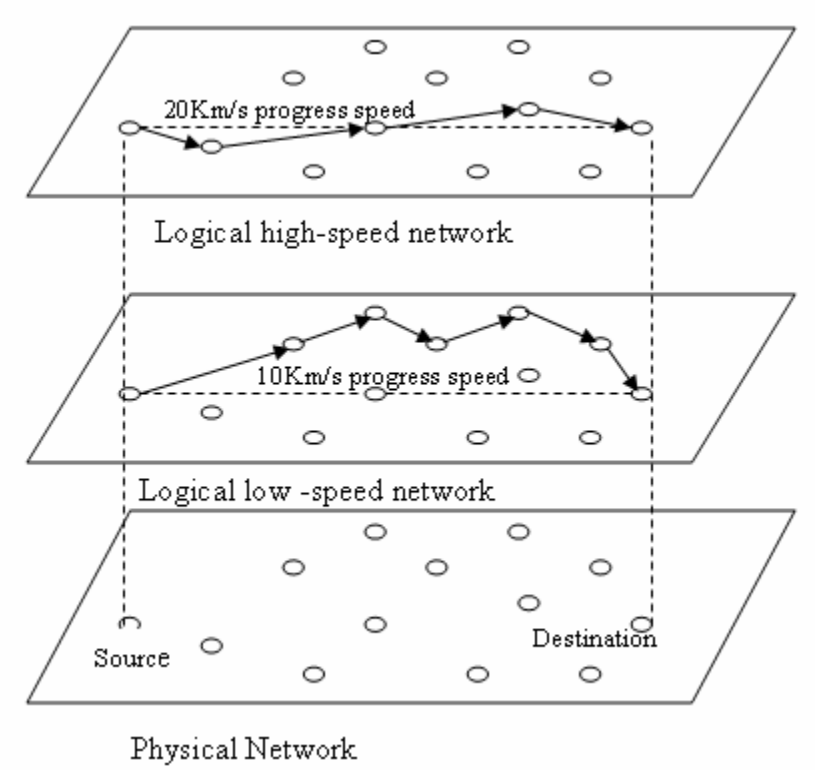


Fig 3.1 Virtual overlay of multiple speed layers

By replicating the single network-wide speed guarantee mechanism which was taken from SPEED routing protocol [2], this protocol provides multiple layers of network-wide speed guarantees, called virtual overlay of multiple SPEED layers on top of a single physical network, for different end-to-end delays as shown in Fig 3.1. For this, each node has the protocol structure as in Fig. 3.2.

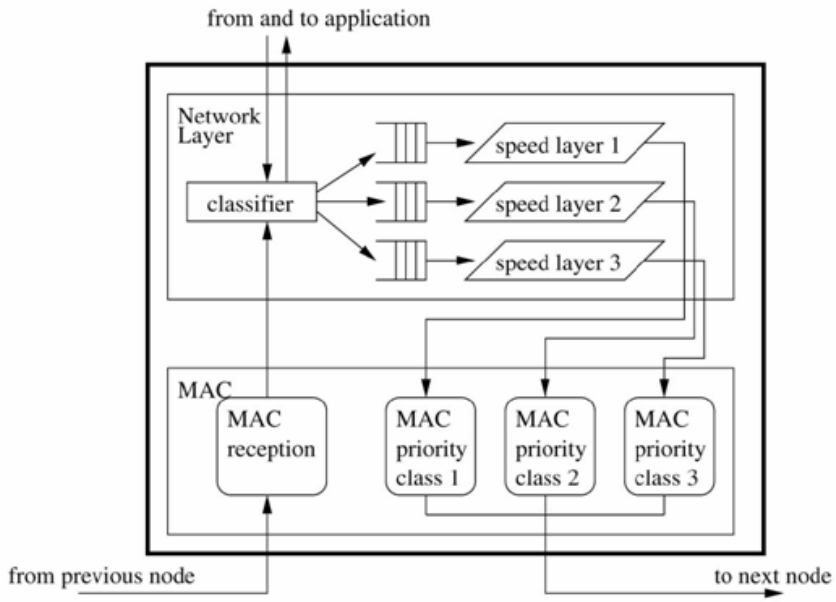


Fig 3.2 Protocol structure of each sensor node

When the packet reaches to the node, the protocol performs:

- Virtual isolation of incoming packets according to their speed classes
- Placing them into appropriate priority queues (as shown in Fig. 3.2.)
- Each speed layer L independently runs the above SPEED mechanism (shown on Fig 3.1) by selecting from its queue
- The packets in the highest speed queue is served in first-come first-served (FCFS) discipline, followed by the next highest speed queue, and so on.

As can be seen from Fig 3.2, a special support from MAC layer is needed for packet prioritization.

3.4.1.2.3 **Reliability Domain**

There exist multiple redundant paths to the final destination in a dense sensor network even though they may not be of the shortest paths and non shortest path is sometimes acceptable as long as it can deliver a packet within the end-to-end deadline. This protocol exploits such inherent redundancies to guarantee the required end-to-end reliability level of a packet delivery. The probability that the packet

reaches its final destination grows as the number of paths used to deliver a packet increases through out the network, despite of packet drops, node failures, packet delay and errors on wireless links. Fig. 3.3 shows how a low reliability packet is delivered using a single path and a high reliability packet is delivered over multiple paths and sees the effects when the path breakage happens.

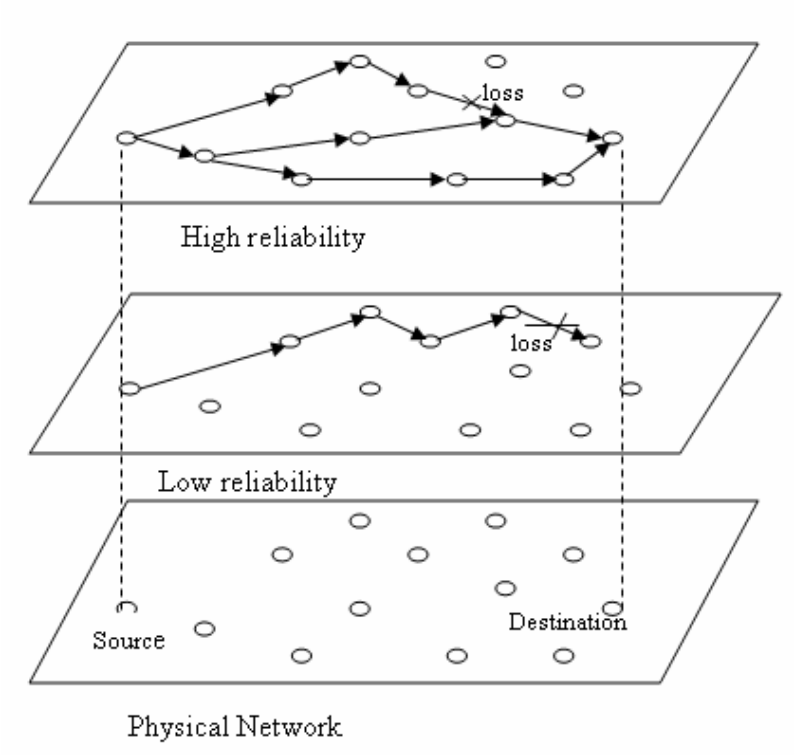


Fig 3.3 Service differentiation in reliability domain

3.4.2 **LEACH** (Wendi Rabiner Heinzelman et al., 14), (Dezhen Song, 20), (K. Dasgupta et al., 31), (S. Bandyopadhyay et al., 46)

3.4.2.1 **Introduction**

LEACH is a self-organizing, adaptive clustering routing protocol which uses randomization to distribute the energy load evenly among the sensors in the network and use clustering technique to utilize energy more effectively. In LEACH, the nodes organize themselves into local clusters within their radio ranges and topology, with one node acting as the local and temporary BS or *cluster-head* includes randomized rotation of the high-energy cluster-head position such that it rotates among the various sensors in order not to drain the battery of a single sensor node. In addition, LEACH performs

local data fusion or aggregation to "compress" the amount of data being sent from the clusters to the BS to reduce energy dissipation and enhancing system lifetime.

3.4.2.2 **Algorithm Details**

The operation of the algorithm is broken up into repetitive *rounds*, where each round begins with a set-up phase and followed by a steady-state phase. Detail discussion is as follows.

3.4.2.2.1 **Setup phase**

The setup phase is divided into three main sub-phases as:

- *Advertisement phase,*
- *Cluster setup phase, and*
- *Broadcast schedule*

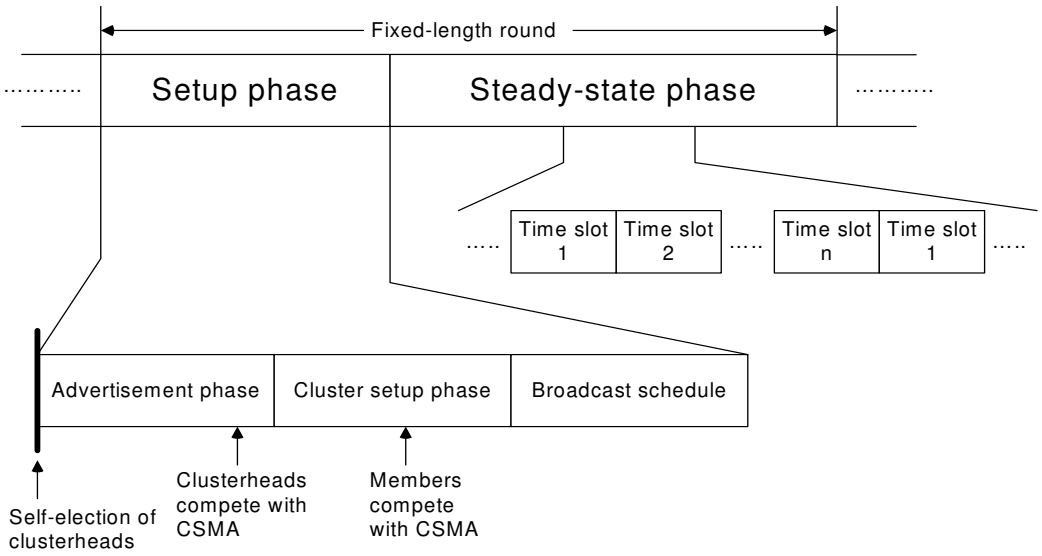


Fig 3.4 Leach rounds

❖ **Advertisement Phase:** At the start, each node decides whether or not to become a cluster-head for the current round when a cluster is formed based on the suggested percentage of cluster heads for the network and the number of times the node has been a cluster-head so far. Then the node selected as a cluster-head for the current round broadcasts an advertisement message to the rest of

the nodes. The non-cluster-head nodes must keep their receivers on during this phase of set-up to hear the advertisements of all the cluster-head nodes. After this phase is complete, each node decides on its own solely on the basis of signal strength.

- **Cluster Setup Phase:** After each node has decided to which cluster it belongs based on the received signal strength, it must inform the cluster-head node that it will be a member of the cluster. During this phase, all cluster-head nodes must keep their receivers on to receive the signal with in the cluster based on signal strength.

- **Schedule Creation:** The cluster-head node which receives all the messages for nodes that would like to be included in the cluster, creates a time division multiple access (TDMA) schedule telling each node when it can transmit. This schedule is broadcast back to the nodes in the cluster.

3.4.2.2.2 **Steady-state phase**

Once a setup phase is completed, then data transmission phase will commence.

- **Data Transmission:** Once the clusters are created and the TDMA schedule is fixed, nodes with in the cluster start sending data during their allocated transmission time to the cluster head. The radio of each non-cluster-head node can be turned off until the node's allocated transmission time is reached, thus minimizing energy dissipation in these nodes. When all the data has been received, the cluster head node performs signal processing or data aggregation functions (minima, maxima, average etc) to compress the data into a single signal and sent to the BS. Since the BS is far away, this is a high-energy transmission than sending data to the cluster-head from the members of the cluster, which consumes less energy. This is the steady-state operation of LEACH networks. After a certain time, which is determined a priori, the next round begins with each node determining if it should be a cluster-head for this round and advertising this information.

3.4.3 **Directed Diffusion** (C. Intanagonwiwat et al., 3), (Intanagonwiwat C. et al., 11)

3.4.3.1 **Introduction**

Directed diffusion is a scalable; query based routing protocol which uses a diffusion or flooding technique for data/packet dissemination to the network. It is location dependent and data-centric,

where Data is *named* using attribute-value pairs. The protocol starts sending a sensing task from the BS to all the nodes throughout the sensor network as an *interest* for named data. This dissemination sets up *gradients* within the network if the sensing **data** matching the interest. Events start flowing towards the originators of interests along multiple paths. The sensor network *reinforces* one or a small number of these paths as positive and reinforces others as negative not to wait for data through those paths.

3.4.3.2 **Algorithm details**

The algorithm has many stages to accomplish the task. The detailed steps for the algorithm are as follows.

3.4.3.2.1 **Naming:** In directed diffusion, every task is *named* by, for example, with a list of attribute-value pairs that describe a specific task. Such a task description is called an *interest* which was sent from the BS. As an example the car tracking task might be described as:

- **type** = *car* // Type of sensing/detecting
- **interval** = *20 ms* // Event send back time
- **duration** = *10 seconds* // For how long will it work
- **rect** = *[100,100, 200, 400]* // sensing /detecting area

The data sent in response to interests are also named using a similar naming scheme. Thus, for example, a sensor that detects a car might generate the following data and sent to the BS:

- **type** = *car* // type of car seen
- **instance =** *military convoy* // instance of this type
- **location** = *[125, 220]* // node location
- **intensity** = *0.6* // signal amplitude measure
- **confidence** = *0.85* // confidence in the match
- **timestamp** = *01:20:40* // event generation time

Given a set of tasks supported by a sensor network, then, selecting a naming scheme is the first step in designing directed diffusion for the network.

3.4.3.2.2 **Interest propagation:** The named task description, as shown above, in a group constitutes an *interest* for the given application. An interest is usually injected into the network from the BS, the sink. This sink node records the task, and for each active task, then it periodically *broadcasts an* interest message to each of its neighbors.

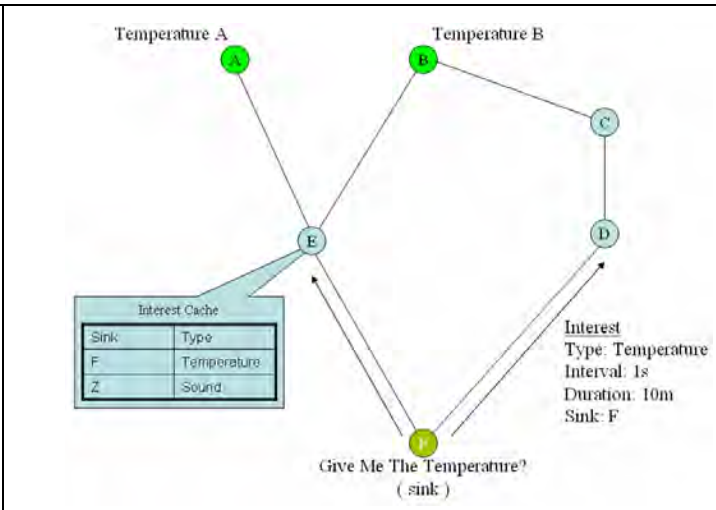
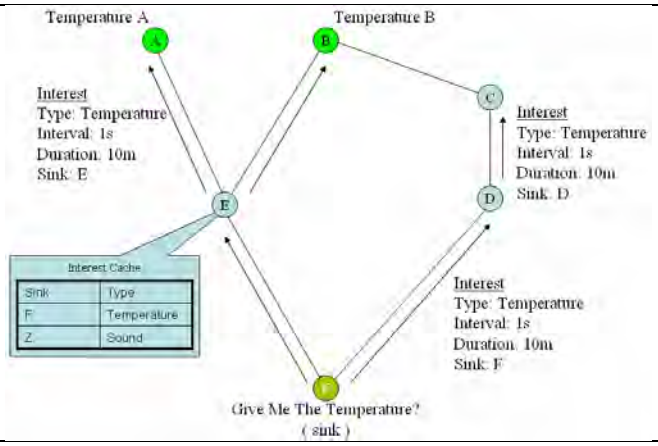
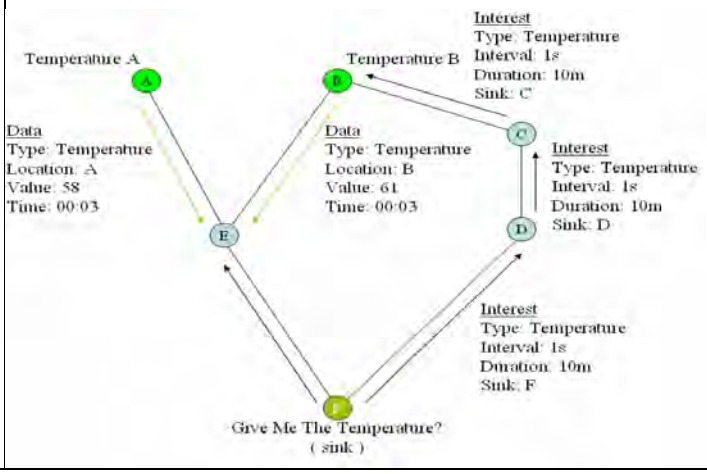
3.4.3.2.3 **Gradient formation:** A gradient specifies a *value* and a direction for the foreseen data from the sensors. When a node receives an interest, it checks to see if the interest exists in the cache. If no matching entry exists, the node creates an interest entry and form a gradient to the neighbor that it received. If there exists an interest entry, but no gradient for the sender of the interest, the node adds a gradient with the specified value. It also updates the entry's timestamp and duration fields appropriately. Finally, if there exists both an entry *and* a gradient, the node simply updates the timestamp and duration fields.

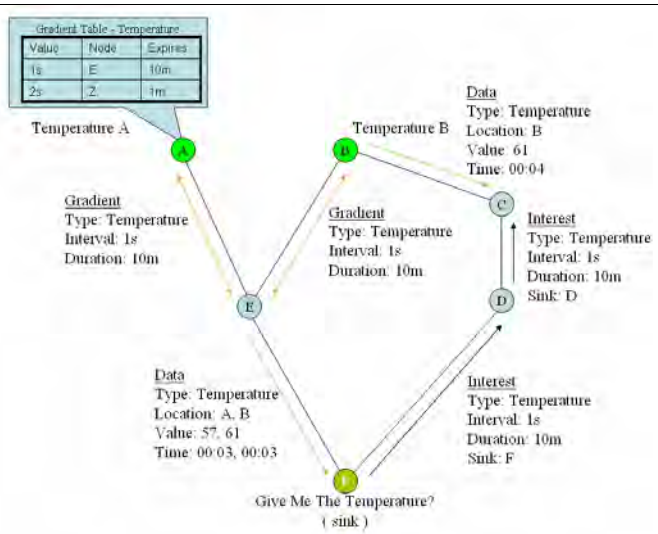
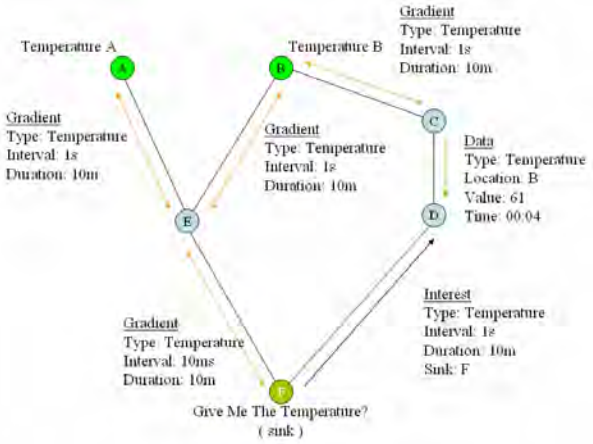
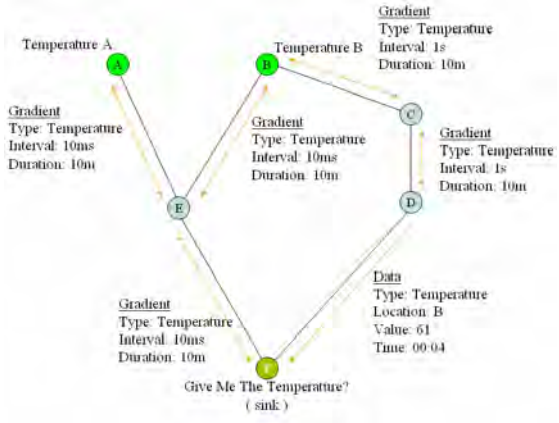
3.4.3.2.4 **Data Propagation:** For a sensor node with in a specified rectangle, when it finds one, interest matching with sensing data, it computes the highest requested event rate among all its outgoing gradients. The node tasks its sensor subsystem to generate event samples at this highest data rate with the specified attributes and sends to each neighbor for whom it has a gradient, an event description every second.

3.4.3.2.5 **Reinforcement:** Once sources detect a matching target, they send low-rate events, possibly along multiple paths, towards the sink. After the sink starts receiving these low data rate events, it *reinforces* one particular neighbor in order to get a high quality data rate with given interval of time. To reinforce this neighbor, the sink re-sends the original interest message but with a smaller interval, i.e. higher data rate.

The following series of figures (Table 3.1) show the steps of the algorithm.

Steps followed for directed diffusion protocol

The sink initiates the Interest to all the neighbours (asking temperature)	
The interest diffuses to the neighbours	
Data propagation (from sensor A)	

Gradient formation for sensor A	
Gradient formation for sensor B in addition to sensor A.	
Gradient for both A and B with the direction of E.	

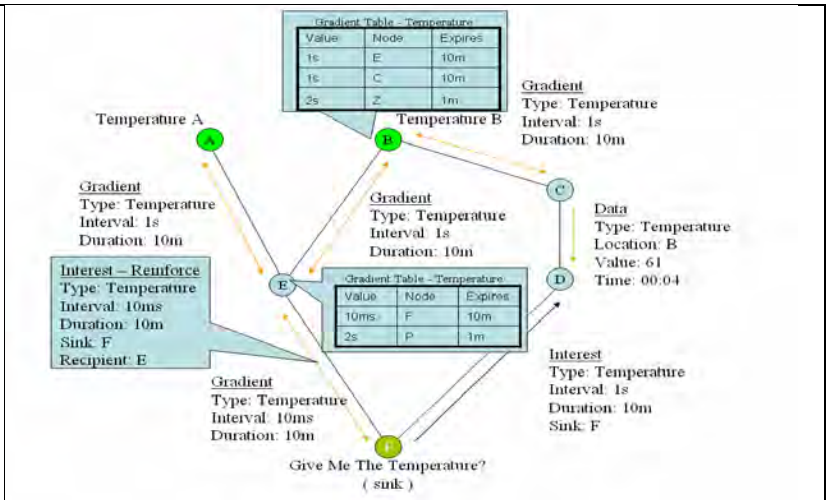
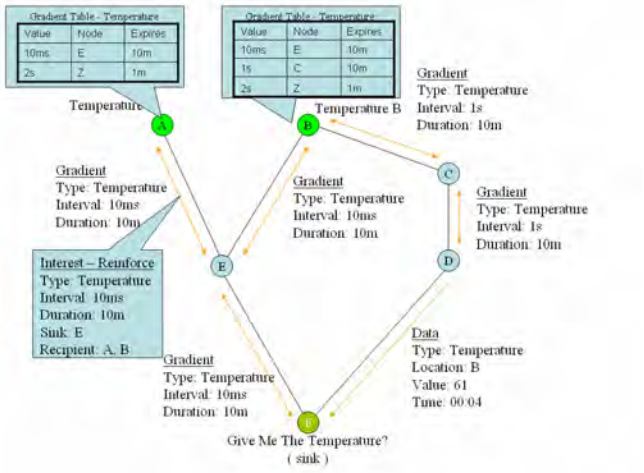
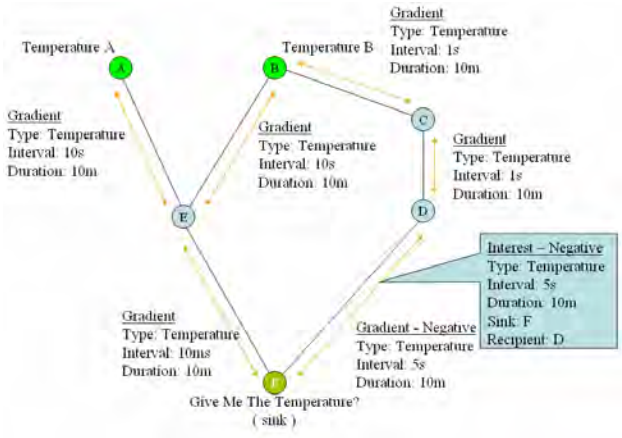
Positive reinforcement for path E.	
Positive reinforcement for the path	
Negative reinforcement for the delayed path	

Table 3.1 diagrammatic representation of the directed diffusion routing protocol

3.4.4 **AODV** (C. Perkins et al., 9), (Charles E. Perkins, 19), (Elizabeth M. Royer et al., 32)

3.4.4.1 **Introduction**

AODV is a dynamic, self-starting, multihop routing protocol between participating mobile nodes wishing to establish and maintain an ad hoc network primarily designed for MANET. It is a reactive routing protocol; therefore, routes are determined only when needed.

In AODV the messages used are route-request (RREQ), route-reply (RREP), route-error (RERR) and hello messages. The RREQ message is sent whenever required to create a path to the neighbor and the host which receives the RREQ will sent a RREP to the originator. When the source receives the RREP, it records the route to the destination and can begin sending data. If multiple RREPs are received by the source, the route with the shortest hop count is chosen (Sanjit Biswas et al., 27). Hello messages may be used to detect and monitor links to neighbors and if there is a path failure, the RERR message will be notified.

3.4.4.2 **Algorithm details**

The main parts of the algorithm are classified into: route request, route reply, and route request error and hello messages. The discussion is given below.

3.4.4.2.1 **RREQ**

The RREQ message is sent when either the host does not know the route to the needed destination host or the existing route has expired for some reason. To do this the RREQ message includes the destination sequence number which is the last known sequence number of the destination host entry found in the routing table. If it contains no entry for the destination host, then the unknown sequence number flag must be set. The RREQ message also contains the requesting hosts sequence number, which must be incremented beforehand.

3.4.4.2.2 **RREP**

The host can generate the RREP message if the destination is the host itself or if the route to the destination is valid and has the same or greater destination sequence number. When generating the RREP message, the host, if it is not the destination, copies the destination address and the requested

host's sequence number to the corresponding RREP message's fields. If the receiver is the destination host then its own sequence number is incremented and copied to the destination sequence number field. In addition, the hop count is set to zero and the lifetime field of the RREP message is set to the initial timeout value of the host.

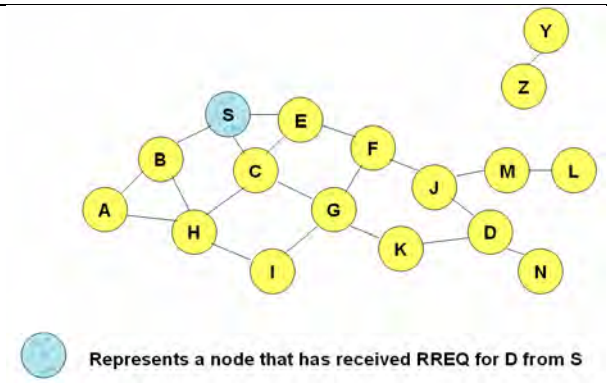
3.4.4.2.3 **RERR**

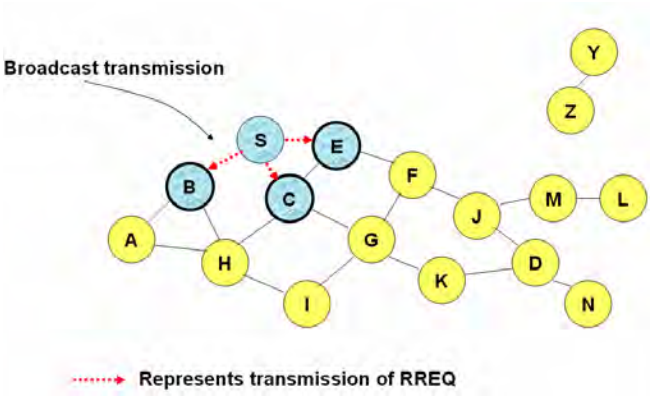
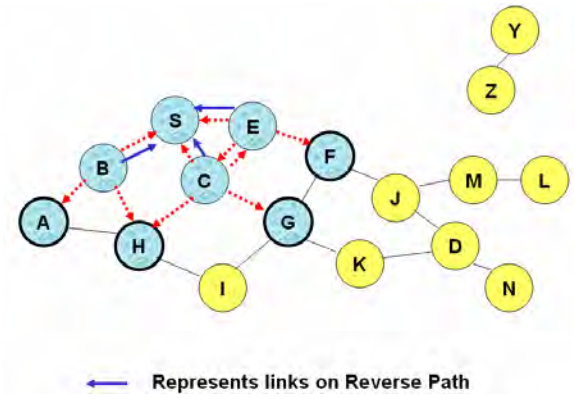
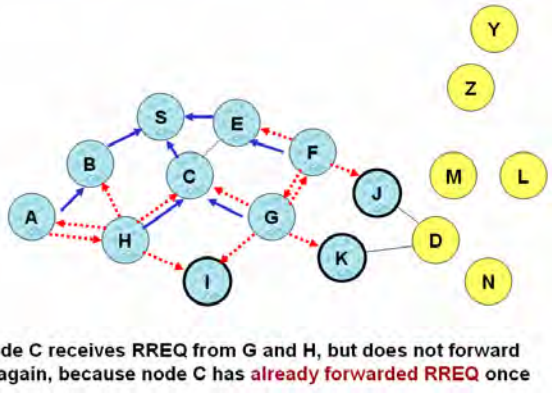
When the link breakage occurs the host must exclude the existing route in the routing table entry by identifying which neighbours can be affected with this breakage and search for another path. Finally the host must unicast the RERR message to the corresponding neighbour or broadcasted if there are many neighbours who need that information.

3.4.4.2.4 **Hello messages**

Hello messages are broadcasted with TTL (Time-to-live) equals to 1, to its neighbors to check whether the link in alive or not so that the message will not be forwarded further. When host receives the Hello message it will update the lifetime of the host information in the routing table. If the host does not get information from the host's neighbour for specified amount of time, then the routing information in the routing table is marked as lost and generates RRER message to inform other hosts of the link breakage.

The following Table (series of diagrams) shows how the AODV protocol works.

Route request from S (Source) to D (Destination)	

S starts broadcasting to all of its neighbours	
Neighbours reply to the request and forward the request to their neighbors if they are not the intended destination	
Reverse path setup for the source from the temporary destination	

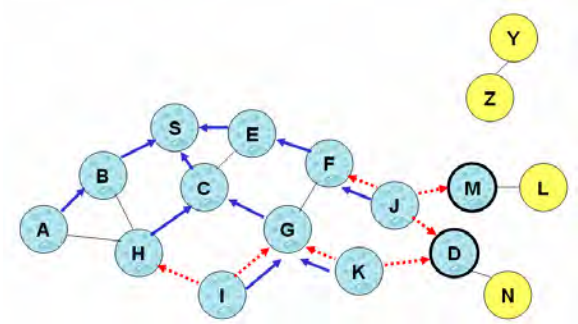
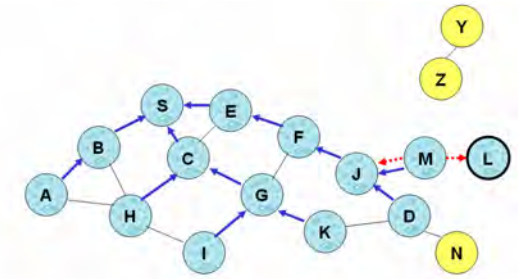
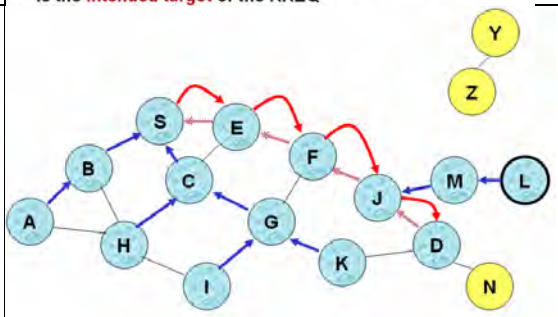
Reverse path setup continues until it reaches to the destination	
Reverse path setup until destination is reached	 • Node D does not forward RREQ, because node D is the intended target of the RREQ
Forward path setup from source to destination	

Table 3.2 diagrammatic representation of the AODV routing protocol

3.4.5 **APS** (Dragos Niculescu et al., 8), [32], (Falko Dressler et al., 39), (S. Capkun et al., 44)

3.4.5.1 **Introduction**

APS is a relative positioning system algorithm, without the use of GPS system for the nodes to determine their location. For this ***landmark*** or ***beacons*** (which know their location either by using

GPS system or other means) help others to calculate and identify their position. A random node in the network will be able to localize itself by estimating its distance to the well known positions, landmarks, of closest BSs.

The key features of this approach are that it is decentralized; it does not need special infrastructure, and provides absolute positioning. The disadvantages are that when the reference moves, positions have to be recomputed for nodes that have not moved.

3.4.5.2 ***Algorithm details***

The algorithm uses two different types of techniques for the nodes to determine their positions. These are (Dragos Niculescu et al., 12):

- *DV-Hop propagation method and*
- *DV-Distance propagation method*

3.4.5.2.1 ***"DV-Hop" propagation method***

This is the most basic scheme for the hops to calculate their location based on the landmarks. Each node in a network maintains a table of the form $\{X_i, Y_i, h_i\}$, where X_i and Yi are X & Y coordinates of node i and h_i is number of hops between node i and nearest beacon, and exchanges updates only with its neighbors with in some time interval.

Once a landmark gets distances to other landmarks, it estimates an average size for one hop, which is then deployed as a correction to the entire network. When receiving the correction, an arbitrary node may then have estimate distances to landmarks, in meters, which can be used to perform the triangulation. The correction, C_i, a landmark at position (X_i, Y_i) computes is:

$$C_i = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_i}, i \neq j, \textit{all landmarks } j \textit{, and } h_i \textit{ is number of hops}$$

A regular node gets an update from one of the three landmarks, and it is usually selected from the closest one, depending on the deployment policy and the time the correction phase of the algorithm

starts at each landmark. Corrections are distributed by controlled flooding mechanism, and drop all the subsequent ones once it sent. This policy ensures that most nodes will receive only one correction, from the closest landmark. When networks are large, a method to reduce signaling would be to set a time-to-live field for propagation packets, which would limit the number of landmarks acquired by a node.

In figure 3.6, as an example, nodes L_1, L_2 and L_3 are landmarks. L1 computes the correction $\frac{100+40}{6+2}=17.5$, which is the estimated average size of one hop, in meters. In a similar manner, L_2 computes a correction of $\frac{40+75}{2+5}=16.42$ and L_3 a correction of $\frac{75+100}{6+5}=15.9$.

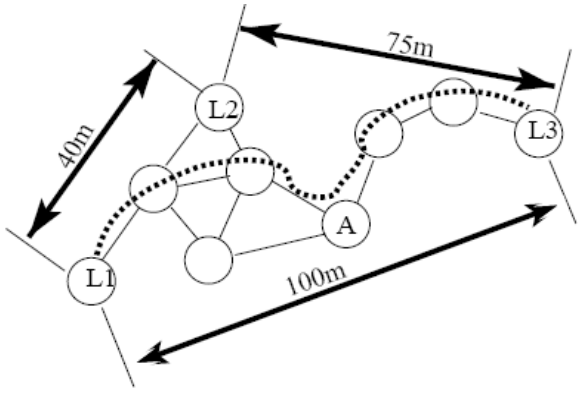


Fig 3.5 How nodes estimate their location using DV-Hop method

In the above example, assume A gets its correction from L_2, its estimate distances to the three landmarks would be:

- L_1= 3X16.42=49.26,
- L_2= 2X16.42=32.84, and
- L_3=3X16.42=49.26.

The advantages of the *"DV-hop"* propagation scheme compared with others are its simplicity and it does not depend on any measurement error.

3.4.2.2 *"DV-distance" propagation method*

This method is mostly similar with the previous one except that the distance between neighboring nodes is measured using radio signal strength and is propagated in meters rather than in counting number of hops. The distance vector algorithm is now using the cumulative traveling distance, in meters. The advantage of this method is less coarse than *"DV-hop"*, because not all hops have the same size, and the drawback is it is sensitive to measurement errors.

CHAPTER 4: THE SIMULATION SOFTWARE (Ahmed Sobeih et al., 18), [30]

4.1 INTRODUCTION

There are many wireless network simulators that are available with different design and architectures [48]. One of these simulators which is designed for dual purposes, i.e. simulation and emulation, is *J-Sim* simulator which is directly support for WSNs. This simulator is developed entirely in Java, and has a component-based compositional network simulation environment.

J-Sim is implemented with two main component-based software architectures, called the *autonomous component architecture (ACA)* and *generalized packet-switched internetworking framework (called INET).* The basic entities in ACA are *components*, which communicate with one another via sending/receiving data at their *ports.* ACA enables new components to be included into J-Sim in a plug-and-play fashion.

4.2 Features of J-Sim

J-sim has many desirable features when compared with other simulators. To mention some:

- It *is* implemented in Java, makes *J-Sim* a truly platform-independent, extensible, and reusable environment.
- It provides a script interface that allows its integration with different script languages such as Perl, Tcl, Jacl or Python.
- It is a dual-language simulation environment like ns-2 in which classes are written in Java and scripts using Tcl/Java (Jacl).
- Only the public classes/methods/fields in Java can be accessed in the Tcl environment instead of exporting explicitly all classes/methods/fields like other simulators, e.g. ns-2.
- J-Sim exhibits good scalability for the memory allocation to carry out simulation of length 1000 is at least two orders of magnitude lower than that in *ns-2* [36].

- The simulator also support emulation, the virtual simulation environment is integrated with a small number of real hardware devices to facilitate performance evaluation of real-life devices in a large-scale, but well-controlled environment.

For all of these reasons, *J-Sim* has been chosen as the simulation framework for this work especially its scalability.

4.3 **Components of the simulation framework** [33], (Hung-ying Tyan, 42)

As mentioned above, J-Sim is a component based simulation environment for WSNs. Here the components and classes defined and implemented in the simulation framework will be discussed briefly.

4.3.1 **Target node**

Target node represents the external environment of the sensing area and transmit signal to the surrounding based on pre-designed target behaviors. In order to realize a ***target node***, the classes implemented in *J-Sim are* as follows:

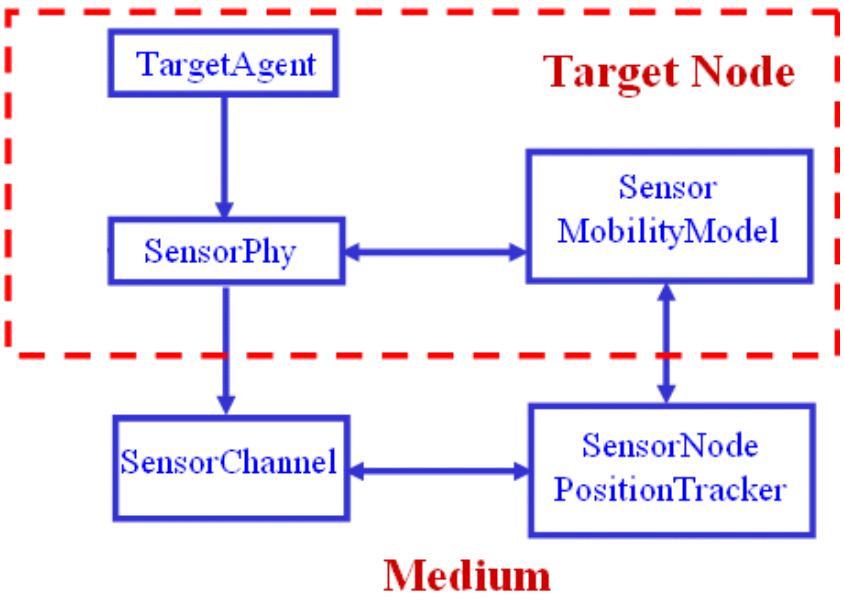


Fig 4.1 Target node internals

TargetAgent: is a class responsible for the target node to periodically generate stimuli (signals) and passes them to the lower layer in order to be transmitted over the sensor channel.

SensorMobilityModel: is responsible for maintaining the location which was specified either in terms of (longitude, latitude, height) or (X, Y, Z), speed and mobility pattern of a target node.

MobilityModel: This is the base class for all mobility models and supports two different mobility models: trajectory-based and random waypoint.

- In a trajectory-based mobility model, a trajectory array provided by the user is used to specify how target and sensor nodes move with a constant speed.
- In a random waypoint mobility model, a mobile node starts from its original location, randomly chooses a location in the simulated area as the destination, and moves in a straight line to that destination location at the speed that is uniformly distributed between 0 and a configurable maximum speed and repeats the same when it reaches the destination.

SensorPhy: implements the sensor physical layer and plays two different roles depending on whether it exists in the protocol stack of a target node or a sensor node. If it exists at a target node, it receive a stimulus generated by *TargetAgent*, query *SensorMobilityModel* to get the up-to-date location of the target node and forward the generated stimulus (together with the location information) to the sensor channel component (*SensorChannel*).

4.3.2 **Sensor Node**

Sensor node is the heart of the simulation framework that is an intermediate between the sink and the target node. In order to realize a sensor node, the classes implemented in *J-Sim* are as follows:

Battery Model: different battery models are designed and implemented as per the real world batteries and with their working principles. These are:

- ***BatteryBase:*** This is an abstract base class for different types of battery models and it defines the ports that are needed for any type of battery model (e.g., to interface with the CPU and radio models).
- ***BatteryTable:*** This class defines a table that specifies the capacity of a battery as a function of its current.
- ***BatteryCoinCell:*** It is a subclass of BatteryBase and implements a Coin Cell battery with the help of its defined table (instance of *BatteryTable*) that specifies the capacity of the battery as a function of its current.
- ***BatterySimple:*** It is also a subclass of *BatteryBase* and implements a simplistic battery model whose capacity is assumed to be always constant (i.e., not a function of the current).
- ***BatteryContract:*** Is responsible for defining the information exchange between the battery model and the CPU and radio models to inform one another about the amount of current that will be drained from the battery, depending on their operational modes.

CPU Model*:* Represents the real model and applications of the CPU. The classes for this model are:

- **CPUBase:** is an abstract base class for a different type of CPU models and it defines the ports that are needed for any type of CPU models (e.g., ports needed to interface with the battery model).
- **CPUAvr:** is a subclass of *CPUBase* and provides values of the current that has to be drained from the battery model in each of the CPU operation modes: *idle*, *sleep*, *off* or *active.*

Radio Model*:* includes the following classes:

- ***RadioBase:*** is an abstract base class for different types of radio models and it defining the ports that are needed for any type of radio models (e.g., ports needed to interface with the battery model).

- ***RadioSimple:*** is a subclass of *RadioBase* and provides values of the current that has to be drained from the battery in each of the radio operation modes: *idle*, *sleep*, *off*, *transmit* or *receive*.

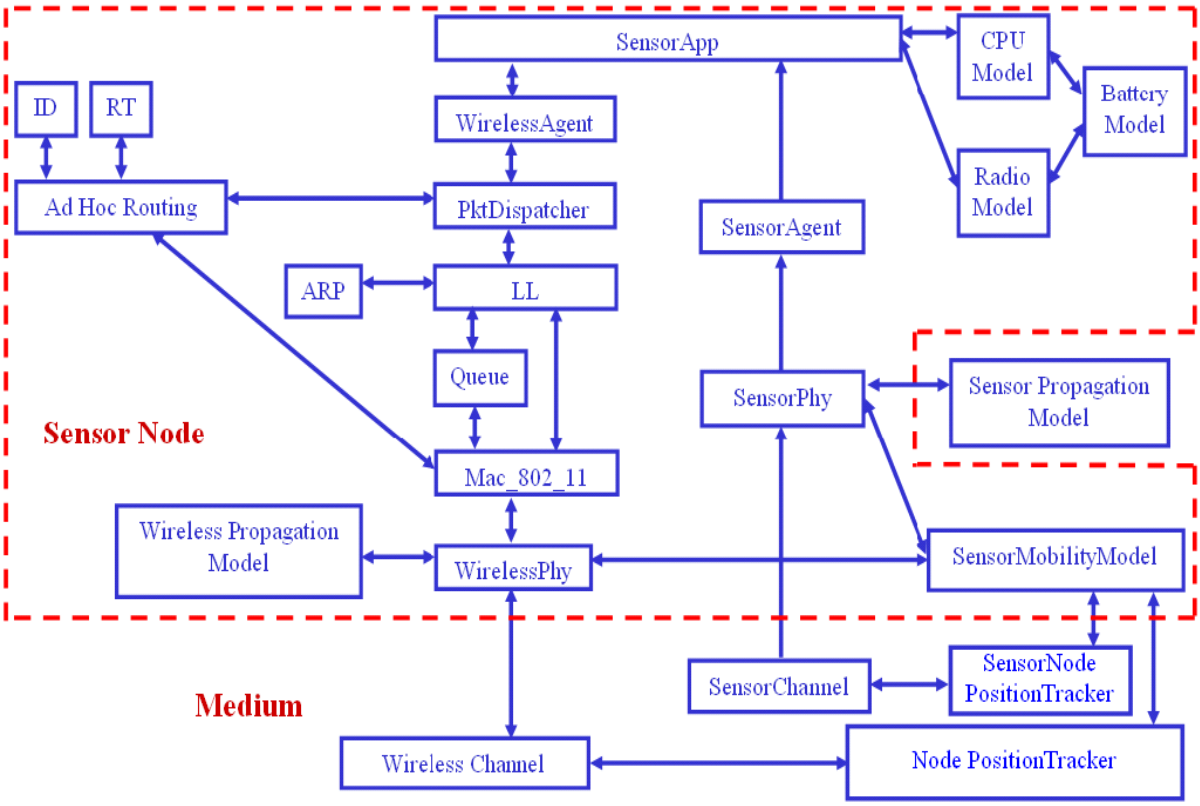


Fig 4.2 Sensor Node internal modules

Sensor Protocol Stack*:* Is part of a sensor node which has the same components with target and communicates to it through these components. These are:

- ***SensorPhy:*** Its role is to receive from the sensor channel a stimulus (signal) generated by a target node, the location of the target node at the time of generating the stimulus and the power with which the stimulus was generated. Then it queries the sensor propagation model to calculate the received signal power (Pr). If Pr is below a certain receiving

threshold, the signal is discarded; otherwise, it is forwarded up to the higher layer in the sensor protocol stack.

- ***SensorPropagationQueryContract:*** It is responsible to define the contract needed for the exchange of information between *SensorPhy* and the sensor propagation model.
- ***SensorAgent:*** This class implements the sensor layer. It receives the stimulus from the lower layer (*SensorPhy*), computes/extracts the application-specific data (e.g., the strength and duration of the stimulus, the signal-to-noise ratio or the location of the target node) and forwards it up to the sensor application layer.

Sensor Application and Transport Layers: include the following classes:

- ***SensorApp:*** This class is responsible for implementing the sensor application layer. It receives the application-specific data from *SensorAgent*, performs certain in-network processing tasks, and passes the resulting data digest down to the transport layer. The digest goes through the wireless protocol stack and will eventually be transmitted over the wireless channel to the sink node.
- ***WirelessAgent:*** It is a subclass of the *Protocol*, a key class for implementing transport protocols, implements a transport layer between the sensor application layer and the wireless protocol stack of a sensor node. *WirelessAgent* receives from *SensorApp* the application-specific data that is to be sent to the sink node, encloses this data in a *SensorPacket* and passes it down to the wireless protocol stack in order to eventually the transmitted over the wireless channel to the sink node.

Wireless Protocol Stack*:* It is built in a plug-and-play fashion using classes that constitute the simulators' wireless network extension. This stack helps to communicate with the BS. These are:

- ***PktDispatcher**:* provides the functionality of the IP layer of the real world; i.e., forwards incoming packets to an appropriate set of output ports connected either to an upper layer protocol or a lower layer component.
- ***ARP:*** implements the address resolution protocol (ARP).

- ***LL:*** It receives IP packets from *PktDispatcher* and then queries *ARP* to find out the MAC address of the next hop to which the IP packet should be forwarded and inserts it in the interface queue of the underlying wireless interface card. Outgoing IP and ARP packets are buffered in the *Queue* component.
- ***Mac 802.11****:* implements the IEEE 802.11 MAC protocol.
- ***WirelessPhy:*** implements functionalities of the physical layer of a wireless card. It queries *WirelessPropagationModel* to determine the received signal power and delivers a data packet only if the received signal power is at least equal to a certain receiving threshold which was assigned a priori.

4.3.3 **Sink Node**

A sink node can also be constructed in a plug-and-play fashion using a sensor application layer (*SensorApp*), a transport layer (*WirelessAgent*), a physical layer (*WirelessPhy*) and a wireless protocol stack in general as explained above (at sensor node).

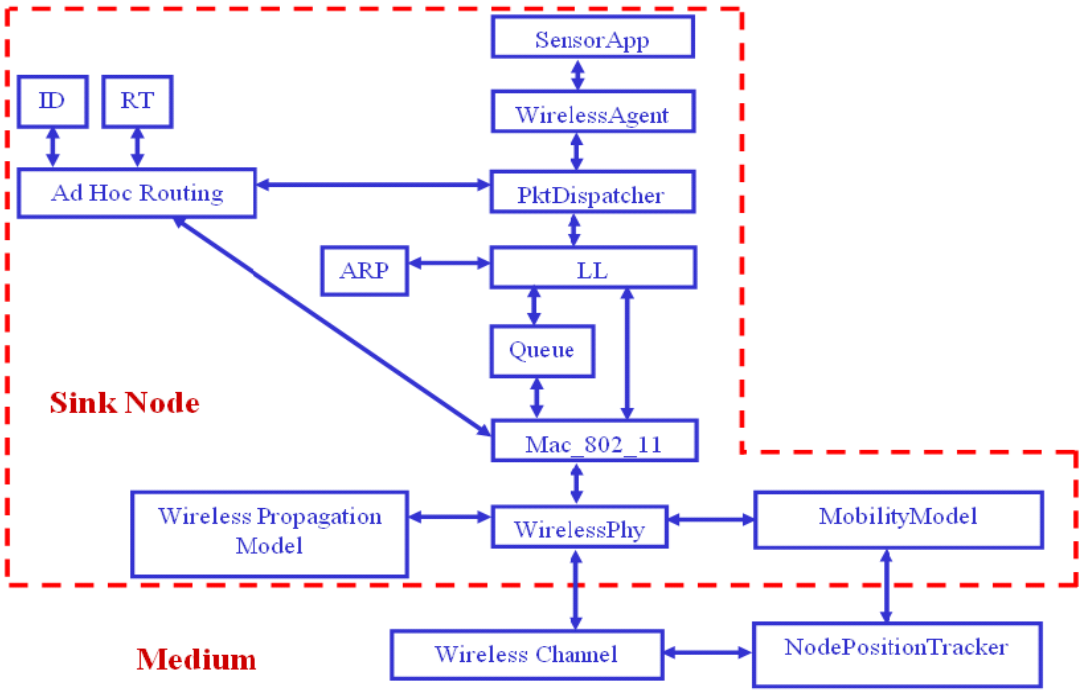


Fig 4.3 Sink node

4.3.4 **Sensor Channel**

In order to realize a ***sensor channel,*** which is common to both target and sensor node, the classes implemented in *J-Sim are* as follows:

SensorNodePositionTracker: A major function performed by this class is to determine which sensor nodes, reported their location by the *SensorMobilityModel* component, are within the sensing radius of a target node and hence, should receive the stimulus generated by that target node.

SensorChannel: implements the sensor channel. The main function of this class is that it receives a stimulus from a target node, queries *SensorNodePositionTracker* to get the list of sensor nodes that are within the sensing radius of that target node, and then sends the generated stimulus to each sensor node that is on the list and repeats the same after a fixed propagation delay.

4.4 **Operation of the simulator**

The operation of the simulator is as follows: The operation is started by generating a stimulus by a target node periodically and propagated over the sensor channel. The neighboring sensor nodes of the target node which are within the sensing radius of the target node will then receive the stimulus over the sensor channel. After receiving the signal, the sensor node compares the received power with a pre-determined receiving threshold value. The calculation of the received signal power is determined by the sensor propagation model used in the model (e.g., seismic or acoustic).

Each sensor node that receives and detects over the sensor channel has to forward its sensing result to one or more sink nodes over the wireless channel either directly or through other nodes to the BS, the sink. Before sending the data depending on the application for which the sensor network is used, a sensor node may either forward data packets as soon as it detects the stimuli,

or process them first (e.g. apply any aggregate function) and then forward processed data (e.g., average, minima or maxima) to the sink node.

Any in-networking processing mechanism can be implemented in the sensor application layer and the coordination between the sensor protocol stack and the wireless protocol stack is done by the sensor application and transport layers.

CHAPTER 5: MODIFYING THE SIMULATOR AND PROTOCOLS IMPLEMENTATION

Even though the simulator has so many valuable features and advantages when compared with other wireless network simulators, it has some limitations to use it as it is for this thesis work. These limitations and problems are:

- It does not have a module or a package for a trace file, which stores the simulation results like delay, throughput etc registered within some interval, to calculate the metrics.
- Even though the simulator supports WSN simulation, it does not have implemented any of the sensor routing protocols except AODV.
- Some of the routing protocols like MMSPEED have special support from the MAC layer. These leads not only implement the protocol but also require changing and adjusting of the MAC layer of the simulator. This is not a simple task to replace the existing MAC layer with the new one since all the modules of the later should be adjusted based on the former to create compatibility with other packages.
- The other problem for these protocols is their power model. They have different power model which were designed by different researchers for CPU and radio models (transmit, receive, idle and sleep mode).

5.1 Addition of features

One of the difficult tasks for this work is to add the metrics' calculation modules to the respected files of the simulator. To do this, it should be understood the packet format for that protocol and the files that are involved either directly or indirectly for packet movement during the simulation so that the effect (change, add or reduce) of each file should be registered to those metrics' modules. The major files which were modified and the codes that are added on the simulator files for the three metrics are as presented below.

- **Average delay**
  - In order to calculate the average end-to-end delay from the source to destination, sensorApp.java, which is the main file of the application layer package, is modified as shown below.

SensorApp.java file

```
      private double avgLatency = 0.0;
      private double sumLatency = 0.0;
      private int latencyCount = 0;

protected synchronized void recvSensorPacket(Object data_)
    {
          SensorPacket spkt = (SensorPacket) data_;

             ...


            else {
                        double latency=getTime()-spkt.getSendTime();

System.out.println("==========LATENCY="+latency+"===========");
                       if (latencyPlotPort.anyOutConnection()) {
                     latencyPlotPort.exportEvent(LATENCY_GRAPH, latency,
null);
            }

          if (latencyCount == 0) {
                sumLatency = latency;
                latencyCount = 1;
            }
            else {
                latencyCount ++;
                sumLatency = sumLatency + latency;
            }
             avgLatency = sumLatency/latencyCount;

                  this.totalINpackets = this.totalINpackets + 1;


          }
...
 }

public double getAvgLatency() {
        return (this.avgLatency);
    }
```

And the Jacl code to call the modified module is:

JACL file

```
proc getDelay { } {
      global sim node_num node sink_id target_node_num

      set delay 0
        #reset variables


      if {[! $sim getTime]>= 500.0 } {
         $sim stop
      for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
  set delay [expr [! n0/app getAvgLatency]]"
      }
}
```

- **Average packet delivery ratio**
  - These are the changes for the simulator's application layer package, sensorApp.java, to calculate the average packet delivery ratio.

SensorApp.java file

```
protected synchronized void recvSensorPacket(Object data_)
    {
          SensorPacket spkt = (SensorPacket) data_;

             ...


            else {
this.totalINpackets = this.totalINpackets + 1;

          }
...
 }
public int getTotalINPackets()
      {
            return (this.totalINpackets);

      }
```

And the Jacl code to call the modified module is:

Jacl file

```
proc getPacket { } {
      global sim node_num node sink_id target_node_num

      #reset variables
      set total_packets 0
      set success 0

      if {[! $sim getTime]>= 500.0 } {
         $sim stop
      for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            set curr_packets [! n$i/app geteID]
            puts "Sensor$i Sent $curr_packets Packets to BS"
            set total_packets [expr $total_packets + $curr_packets]

         }

         puts "Base Station Received:[expr [! n0/app getTotalINPackets]+$p]"
         puts "Total Packets sent from all nodes: $total_packets"
        set success [expr ([! n0/app getTotalINPackets].0 / $total_packets.0)
* 100]
         puts " Average dissipated energy : [expr $average_energy /
      }
}
```

- **Average power dissipated energy**
  - The change for the simulator to calculate the average power dissipated energy from the physical layer package, wirelessPhy.java, is shown below.

WirelessPhy.java file

```
public double getRemainingEnergy()
{
if (plotterPort.anyOutConnection()) {
                    plotterPort.exportEvent(ENERGY_EVENT, new
DoubleObj(em.getEnergy()), null);
                }

            return (em.getEnergy());
```

Jacl file

```
proc getEnergy { } {
```

```
      global sim node_num node sink_id target_node_num

      #reset variables
      Set energy 0
      set average_energy 0

      if {[! $sim getTime]>= 500.0} {
         $sim stop
         for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            set val [! n$i/wphy getRemainingEnergy]
            set average_energy [expr $average_energy + $val]

         }

       set energy  [expr $average_energy / [expr $node_num -
$target_node_num]]

      }
```

5.2 **Implementing the routing protocols**

As mentioned above AODV is the only routing protocol which was implemented within the standard simulator. Therefore in order to evaluate the performance of the proposed routing protocols, both for sensor and ad-hoc networks, the protocols should be ported and implemented into the simulator. The steps followed to port the protocols into the simulator are:

- Identify different classes of files from the protocol package
- Put the files in the respective package of the simulator
- Fixing the errors in the files to make them compatible with the simulator
- Configuring the protocols so as to get recognition with the simulator compiler and other parts of the simulator package.

➢ **Identifying files from the protocol package**

The first step of implementing the routing protocols is to identify different files for different applications and put them in their respective places. These include files for:

- Packet frame format
- Power model
- Mobility model (if any)
- Mac layer packages, etc

- **Putting files into the simulator package**

  Some of the files which come with the protocol may be found in the simulator itself so that it is better to check before putting the files, otherwise it generates an error message when compiling time if their modules and number of arguments were not matched. Some of the files may share the name only and the code is totally different and some will have the same name and some percent of code similarity. This will happen because different protocols use different design strategy and algorithms and come up with different files for the part that makes them differ from others.

5.3 Modifying the MAC layer

The tasks that can be done to adjust the MAC layer of the MMSPEED protocol includes replacing the existing files with the new one and adjust the parameters, number of arguments, and the module itself in the case of different architecture or logic used by the protocol. The steps are as follows:

- Replace the existing components with the new one. These are :
  - Mac802_11_ACK_frame
  - Mac_802_11_CTS_Frame
  - Mac_802_11_frame_control
  - Mac_802_11_data_frame
  - Mac_802_11_packet
  - Mac_802_11_RTS_Frame
  - Mac_802_11_timer
  - Mac_802_11e
  - Macphycontract
  - macTimeoutEvt
  - Radio propagation model
  - Node channel contract
  - Priority queue, etc

- After replacing, adjusting methods/functions which were common to other protocols and layers that may take different number and type of arguments, can take place. So the above files should be adjusted according to the requirements of the simulator framework (above, link, and lower, physical, layers of the protocol stack)

5.4 Modifying the power model

Different routing protocols use different power model techniques for CPU and radio models (which includes transmit, receive, sleep and idle). It is not feasible to apply these different types of power models as to lead a wrong conclusion for the measurement of average dissipated energy. So all the power models should be converted to the same known model and all the files concerning this area were redesigned and coded to this same model.

5.5 Configuring the protocols

After all the above works have been done, the protocols should be configured with all the components and ports in order to cooperate and work together. Specifically the protocol will configure with wireless channel, propagation model, and most of the components of the simulator that work in parallel with the protocol either directly or indirectly.

CHAPTER 6: TESTING AND RESULTS

This chapter presents the testing done on the proposed protocols for the critical condition monitoring applications. The testing is done using the widely used performance metrics which were directly affected by the given application. The chapter also shows the results obtained from the simulations of different protocols. The simulations are done on the modified J-SIM for both manually and randomly deployed sensors and for different percentage of network failure with a random selection of the sensor nodes.

6.1 Testing

There are three metrics selected for the testing of the protocols for this work. The metrics which were selected for this work are affected by factors like number of nodes used in the simulation framework, the simulation time, the delivery ratio, and the topology or deployment used.

6.1.1 Simulation Parameters

Below are the list of parameters that are selected for this work based on the usage by other researchers for such or similar applications and on the simulator capability because as the number of nodes increase, the simulation time becomes so large (up to few days) to take a single scenario [2], [7], [13], (Azzedine Boukerche et al., 17).

Parameter	Value
Number of Nodes	100-500
Simulation Time (s)	500
Simulation Area (mXm)	200x200
Node placement	Uniform/Random
Target Speed (m/s)	30
Radio range (m)	40
Source data Rate (packets/s)	10
Transmit Energy (mW)	14.88
Receive Energy (mW)	12.50
Dissipation in Idle (mW)	12.36
Dissipation in Sleep (mW)	0.016

Table 6.1 simulation parameters

6.1.2 Evaluation Metrics

The metrics used to evaluate the performance of the proposed routing protocols are average packet delivery ratio, average dissipated energy and average end-to-end delay.

Average Packet Delivery Ratio (APDR)

Packet Delivery Ratio (PDR) is the ratio of total number of packets successfully received by the destination nodes to the number of packets sent by the source nodes. Mathematically PDR expressed as:

$$PDR = \frac{Number\ of\ \ Packets\ \text{Re}ceived\ by\ Destination}{Number\ of\ \ Packets\ Sent\ by\ Source}$$

And Average PDR (APDR) is the ratio of the sum of all (PDR) of the total number of simulation scenario (with fixed parameters) to the total number of scenarios. In this work for each metrics 20 runs were conducted, so the APDR is calculated as:

$$APDR = \frac{\sum_{i=1}^{20} PDR_i}{20}$$

This metric gives us an idea of how successful the protocols are in delivering packets to the application layer. A high value of PDF indicates that most of the packets are being delivered to the higher layers and is a good indicator of the protocol performance for critical condition monitoring application by creating a reliable network path for the information to reach from the sensors to the BS.

Average dissipated Energy (ADE)

One of the main criteria for sensors is for how long will they effectively use their power before they die. Because once they die, there is no other means to recharge again if they were deployed at remote locations and do not use a solar cell.

Dissipated energy (DE) for a single scenario is the sum of joules of all nodes to the number of nodes. Mathematically this can be expressed as:

$$DE = \frac{\sum_{i=1}^{n} s_i}{n}, where\ S_i\ is\ energy\ of\ sensor\ i\ \&\ n\ is\ number\ of\ nodes$$

And average dissipated energy for 20 runs, becomes:

$$ADE = \frac{\sum_{i=1}^{20} DE_i}{20}$$

Average end-to-end Delay (AEED)

End-to-end (EED) delay is the average delay between sending the data packet by the source and its receipt at the corresponding receiver (BS) for a single scenario. This includes all the delays caused during route acquisition, buffering and processing at intermediate nodes, and retransmission delays at the MAC layer for a given simulation time. This can be expressed as:

$$EED = \frac{\sum_{i=0}^{n} Time\ \ Packet\ \ \mathrm{Re}ceived_i - Time\ \ Packet\ \ Sent_i}{Total\ \ Number\ \ of\ \ Packets\ \ \mathrm{Re}ceived}, where\ \ n\ is\ number\ \ of\ \ nodes$$

Here also the AEED was taken from 20 runs with the formula:

$$AEED = \frac{\sum_{i=1}^{20} EED_i}{20}$$

The scripts for total simulation (creating of sink, sensor and target nodes, configuring, metrics calculation, etc) is listed at appendix B. The total sum of pages of Jacl files for all protocols is more than 150 pages, I attached only for LEACH, uniformly deployed scenario as a sample format.

6.2 Results

In this section the simulation results are shown with respect to the metrics discussed in the previous section. The section also discusses the results based on the graph.

Simulation Results

Here are the graphs for the calculated results obtained from the simulation of five protocols for both uniformly and randomly deployed topology.

6.2.1 Uniformly deployed scenario

In this scenario, the sensors were uniformly distributed through the simulation area and the BS is located at the left side of the area to represent a real world scenario and the target nodes randomly deployed at the right side of the simulation at start and move to the direction where the sensors deployed.

.

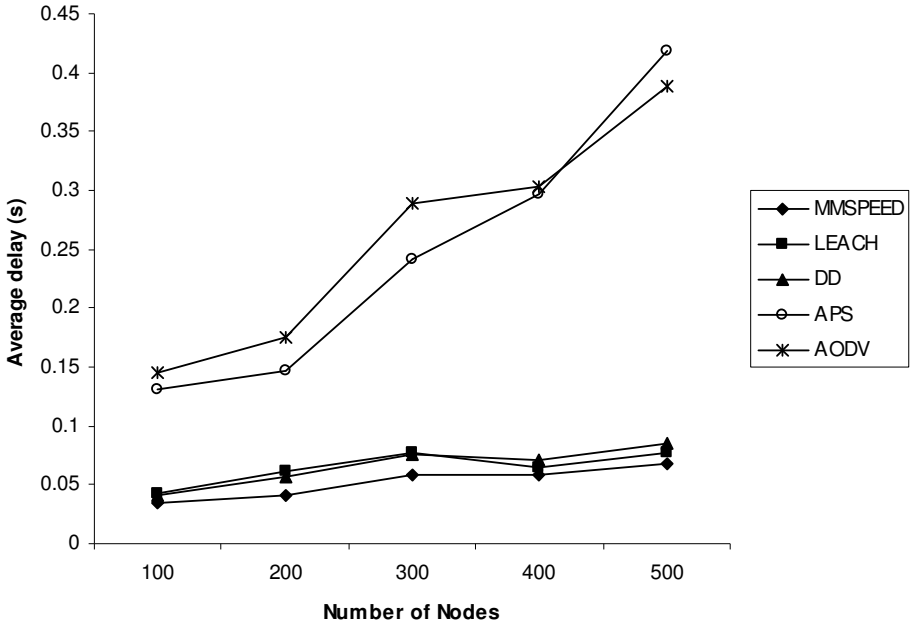


Fig 6.1 Average delay (0 % node failure)

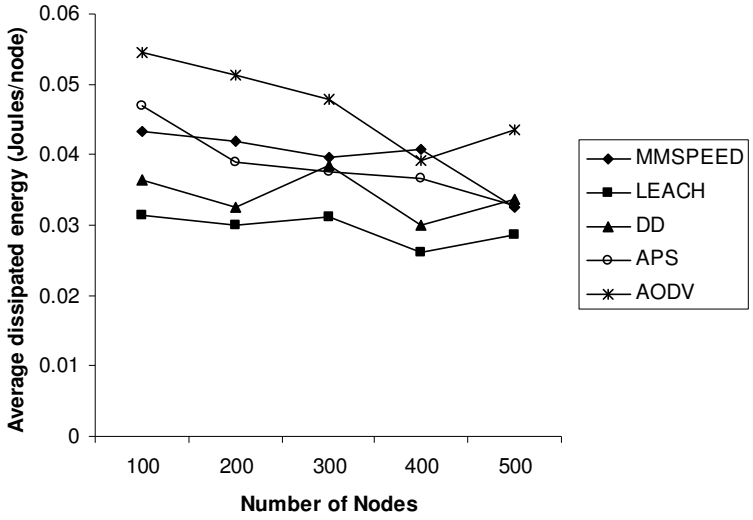


Fig 6.2 Average dissipated energy (0 % node failure)

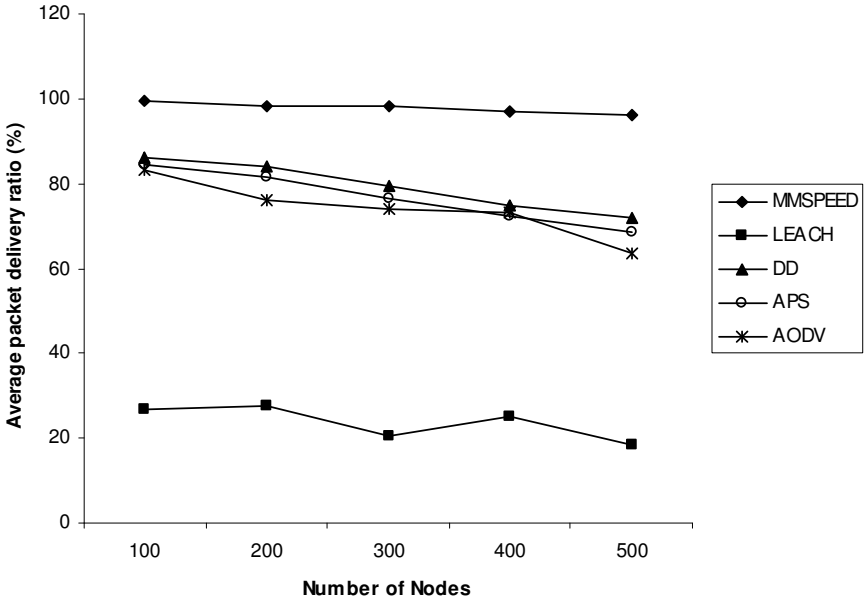


Fig 6.3 Average packet delivery ratio (0 % node failure)

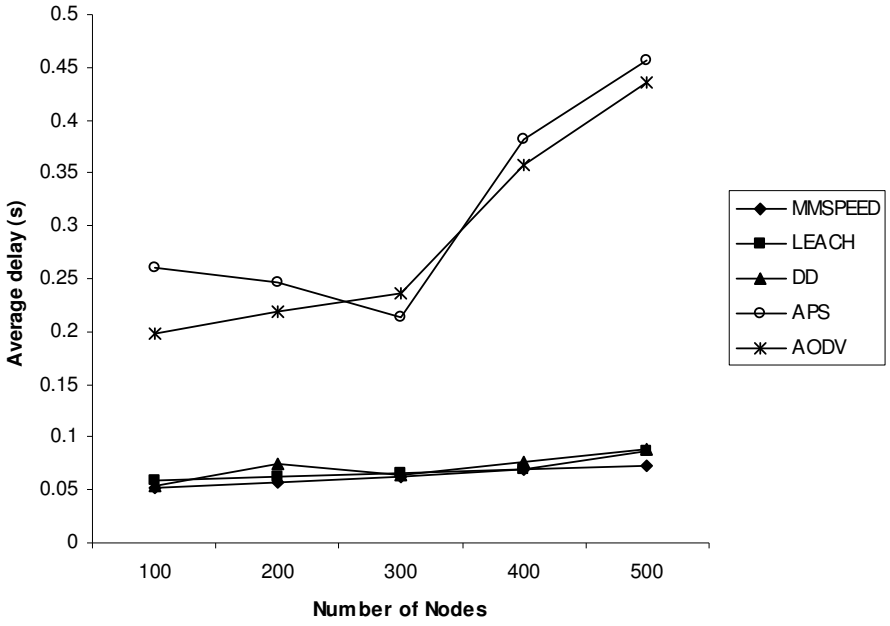


Fig 6.4 Average delay (10 % node failure)

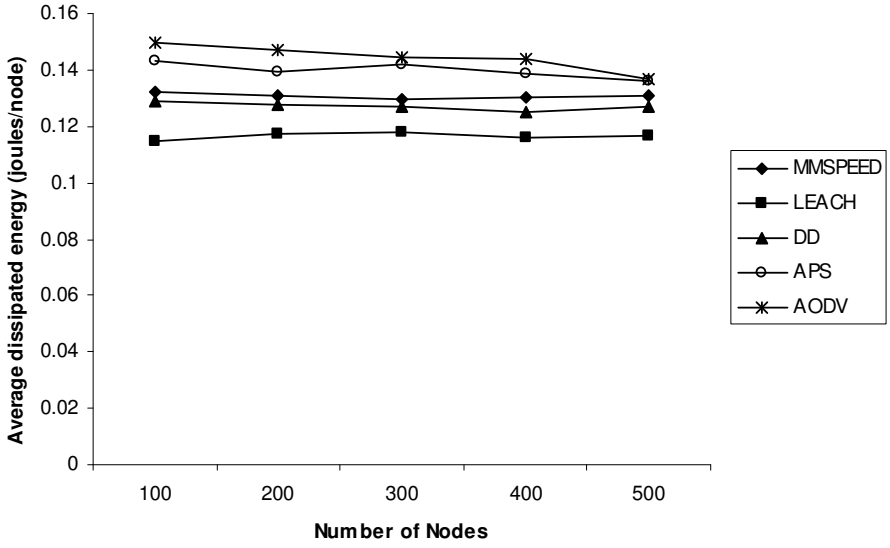


Fig 6.5 Average dissipated energy (10 % node failure)

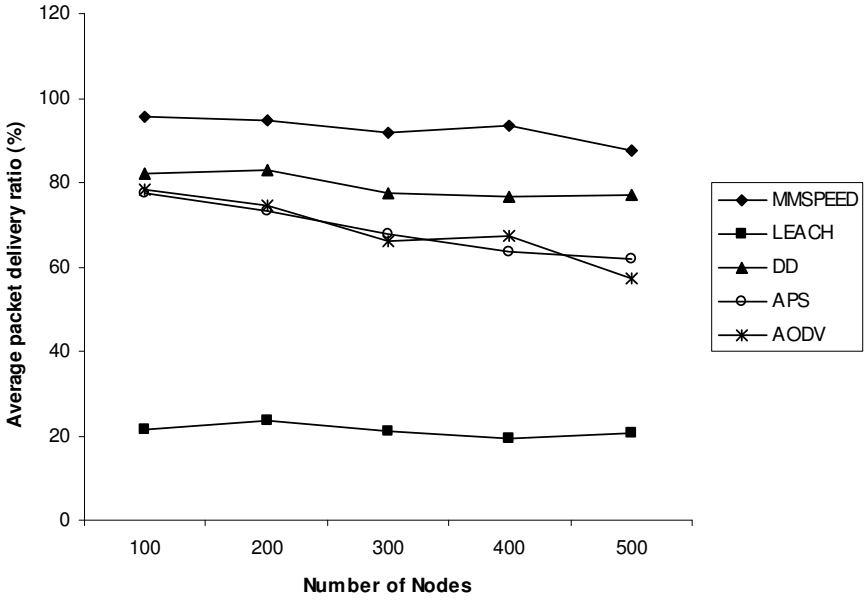


Fig 6.6 Average packet delivery ratio (10 % node failure)

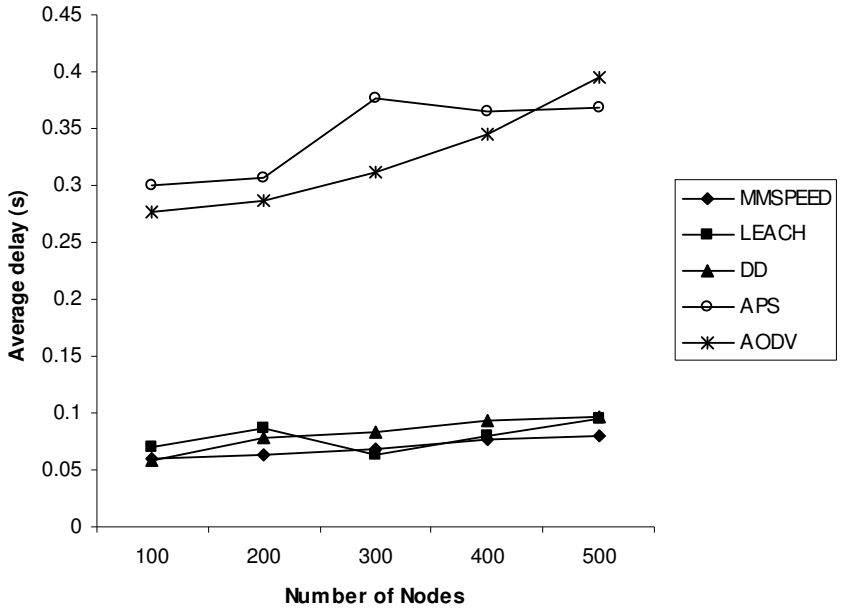


Fig 6.7 Average delay (20 % node failure)

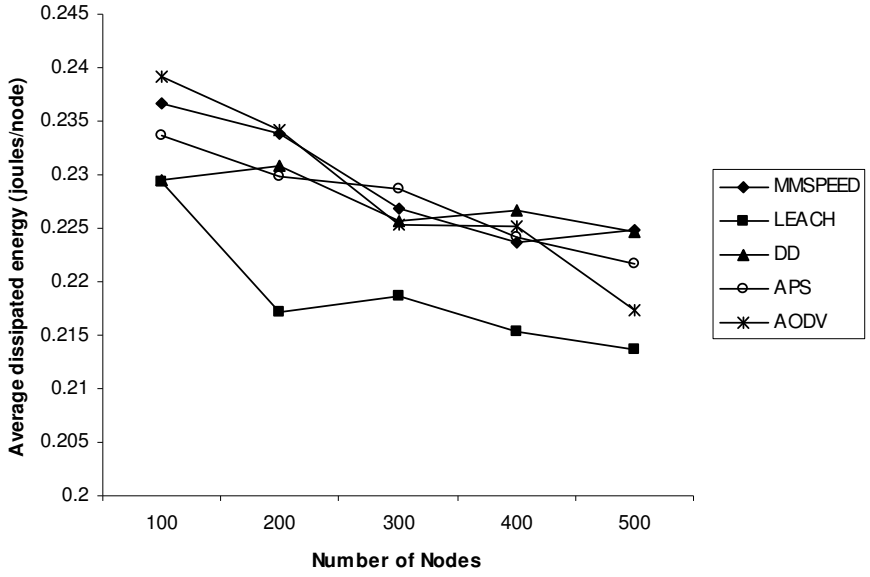


Fig 6.8 Average dissipated energy (20 % node failure)

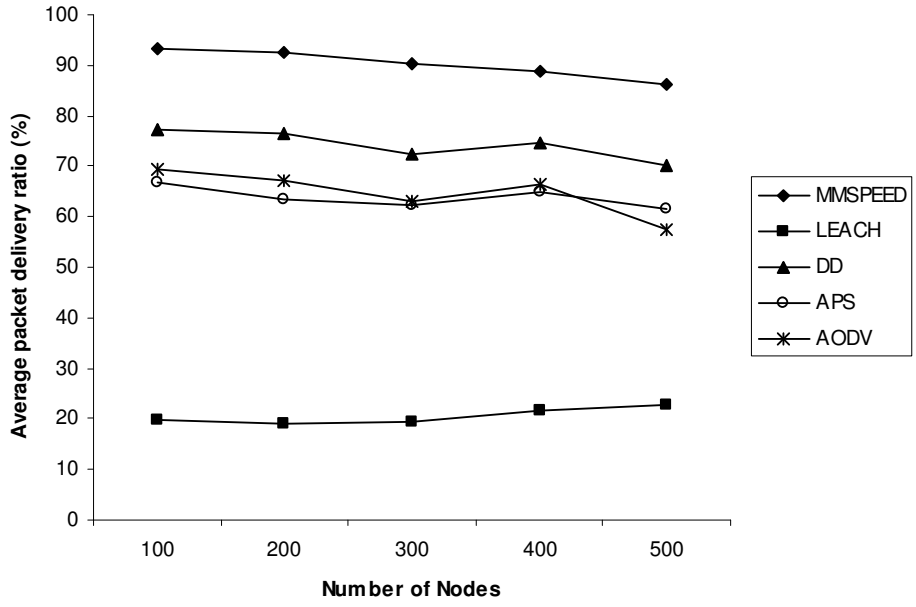


Fig 6.9 Average packet delivery ratio (20 % node failure)

6.2.2 **Randomly deployed sensors**

For this scenario also everything is the same with the above except for deployment, it uses random deployment. In here random numbers for both X & Y coordinates are generated inside the simulation area for nodes location.

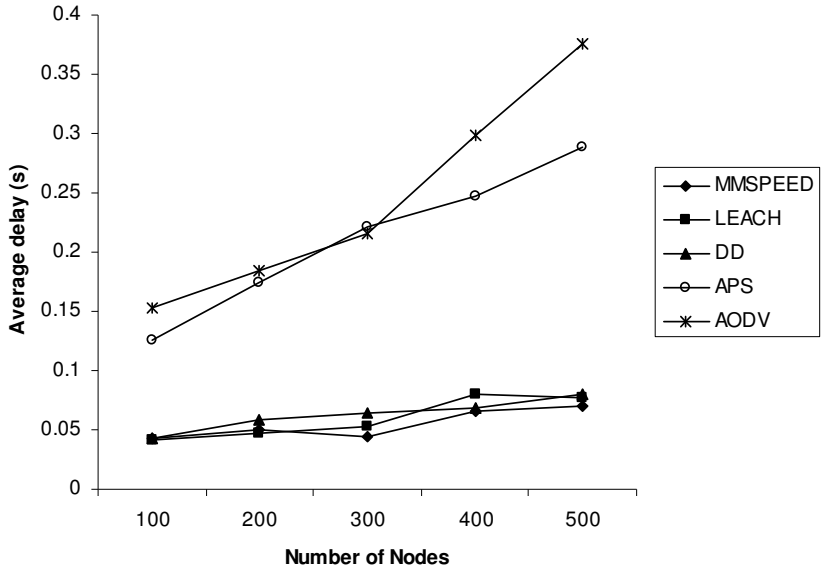


Fig 6.10 average delay (0 % node failure)

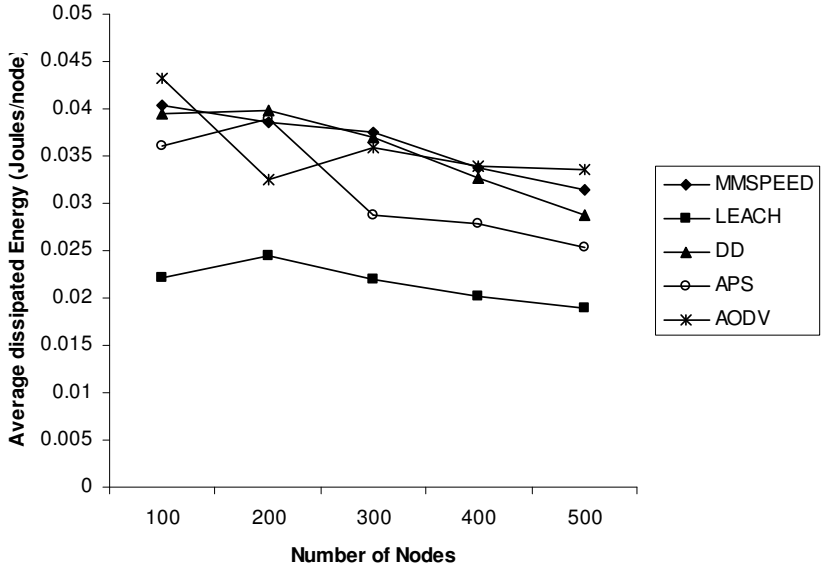


Fig 6.11 Average dissipated energy (0 %node failure)

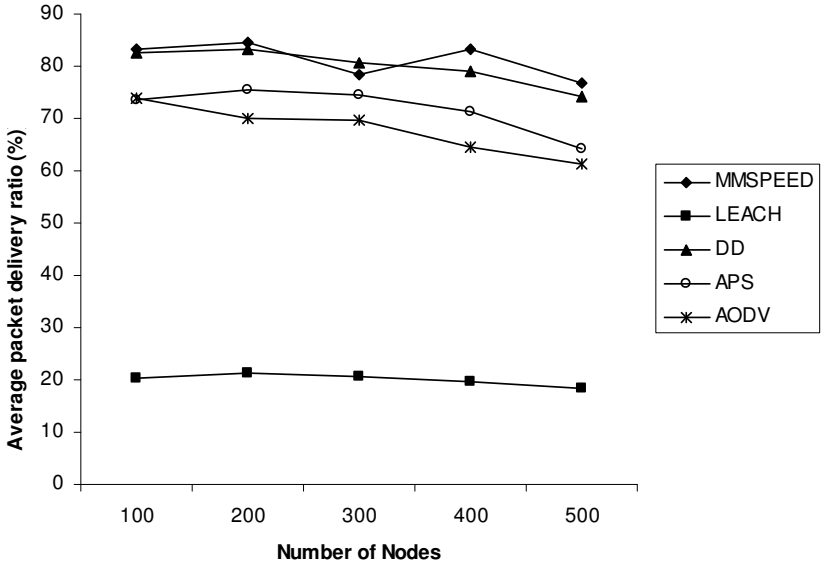


Fig 6.12 Average packet delivery ratio (0 % node failure)

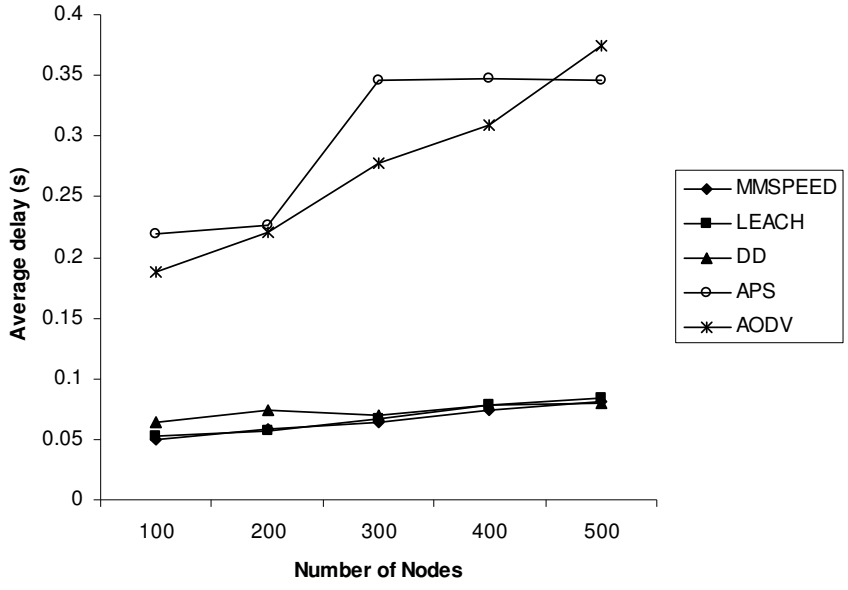


Fig 6.13 Average delay (10 % node failure)

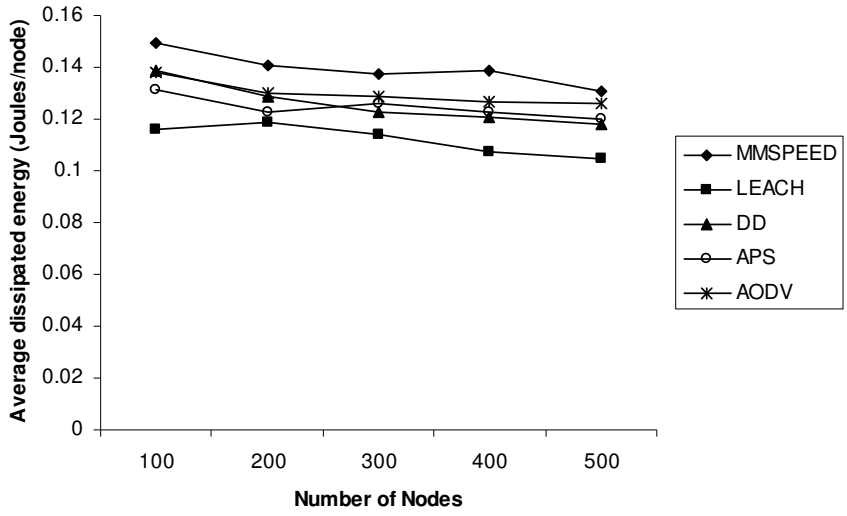


Fig 6.14 Average dissipated energy (10%node failure)

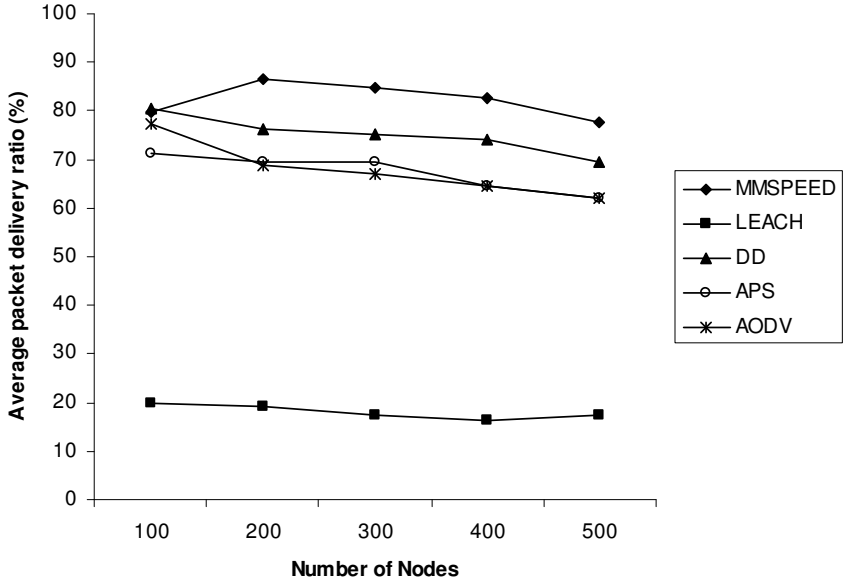


Fig 6.15 Average packet delivery ratio (10 % node failure)

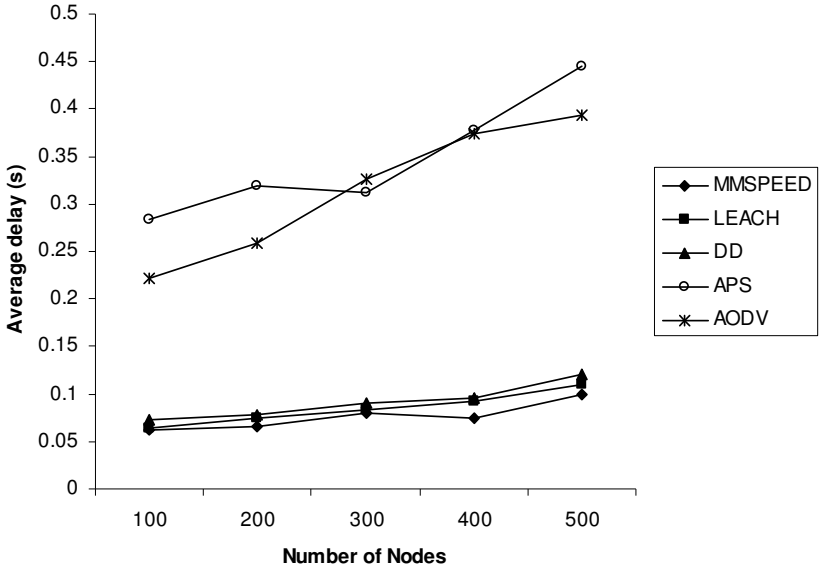


Fig 6.16 Average delay (20 % node failure)

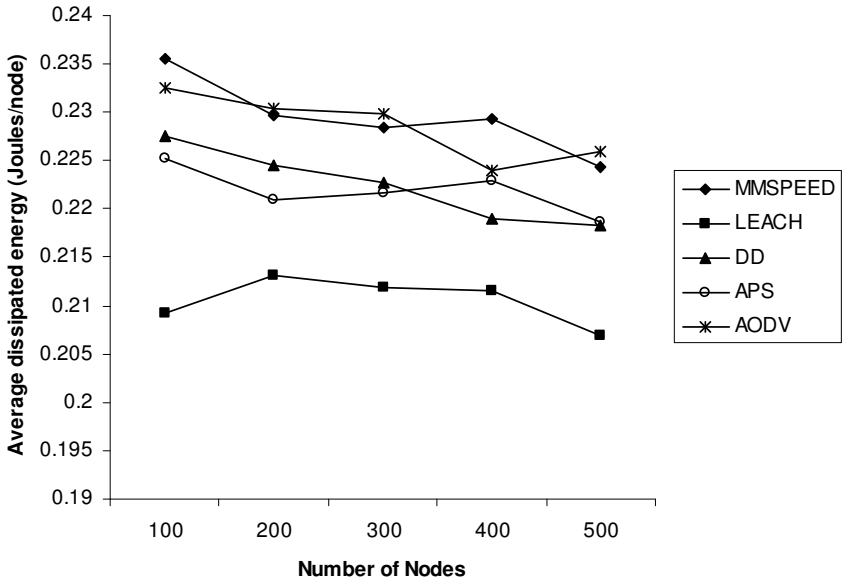


Fig 6.17 Average dissipated energy (20 % node failure)

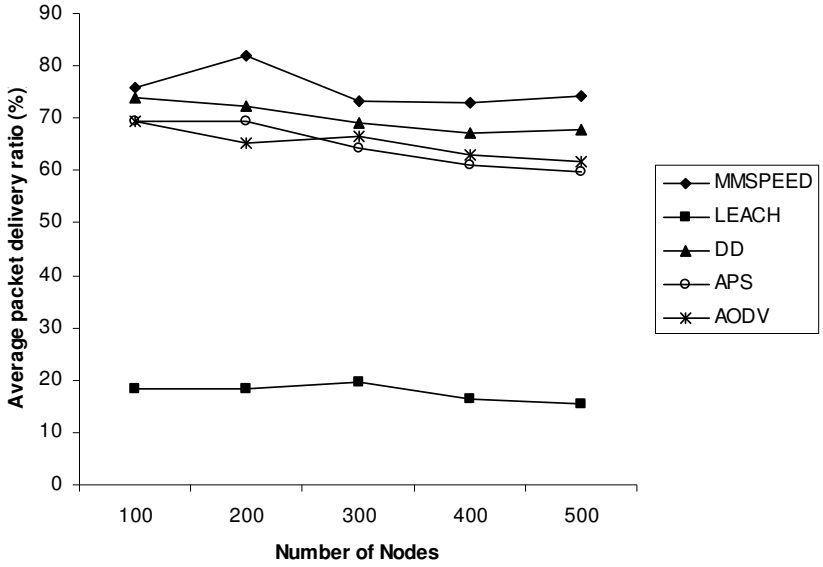


Fig 6.18 Average packet delivery ratio (20% node failure)

6.3 Discussion of Results

The results show that MMSPEED protocol performs better at packet delivery and delay than others and LEACH perform better for energy consumption.

Uniformly deployed scenario

Average end-to-end Delay

Figure 5.1, 5.4 and 5.7 show the average end-to-end delay for 0%, 10% and 20% node failures for a simulation time of 150sec. As can be seen from the graph, MMSPEED has lowest average end-to-end delay for different network status. This is because the protocol uses multiple network-wide speed options for timeline domain so that for various traffics, the packet dynamically chooses a proper speed value from a pre-assigned values depending on its end-to-end deadline.

LEACH and DD have a second low delay than most notably MANET routing protocol AODV and APS. In LEACH, the clusters are the ones to send the packet to the BS so that the delay is minimal and DD has a more or less similar performance of delay with LEACH since the flooding makes the protocol select the lowest delay interval when reinforces the path. But DD uses a

complex mechanism to find a path and the BS sent more messages to setup paths and propagates events, resulting in a more amount of packet collisions, leads more delay than MMSPEED. As can be seen from AODV and APS graph, the route acquisition phase in AODV leads to significant delays for the first few packets, while geographic based routing protocol, APS, doesn't suffer from this even though it there is some delay for the network to configure itself. AODV needs to perform route acquisition repeatedly in order to track the target. In addition, the route discovered through flooding and path reversal has relatively more hops than APS leads to more delay.

Average dissipated energy

Figure 5.2, 5.5 and 5.8 show the average dissipated energy for the three different network statuses as mentioned above. LEACH is a clustering-based routing protocol that minimizes global energy usage by distributing the load to all the nodes at different points in time. Only the cluster head sent data to the BS, makes the protocol to save much amount of energy which were used to transmit all the data sensed with all sensors.

Directed diffusion is data centric in which all nodes in a directed diffusion-based network are application aware. This enables diffusion to achieve energy savings by selecting empirically good paths and by caching and processing data in-network. On the other hand many features of MMSPEED lead to more energy consumption due to more complex computation and longer frame with overhead bits. Periodical "hello message" in AODV lead to more energy consumption while in APS, once deployed and configured, takes less energy consumption for routing.

Average packet delivery ratio

Figure 5.3, 5.6 and 5.9 show the scenario for different status of the network. Because MMSPEED is the one which prepares a reliable path, by creating a multiple path throughout the network, the probability of the packets to reach the BS is very high with a guaranteed delay when compared to other routing protocols. The probability that the packet reaches its final destination

grows as the number of paths used to deliver a packet increases, despite packet drops, node failures, and errors on wireless links.

DD also creates a reliable path with flooding technique from BS to the sensors and select a reliable and error free path for data propagation. This gives the protocol a high performance next to MMSPEED for packet propagation. On the other hand LEACH uses only cluster-heads to send the sensing packet after applying some aggregate function. This means only some packets will reach the BS even though so many packets were sent from the sensors, which reduce the ratio.

AODV and APS only respond to severe congestion, which leads to link failures (i.e., when multiple retransmissions fail at the MAC layer) and reduces the overall packet delivery.

Randomly deployed scenario

The rest of simulation results shown, i.e. from Fig 5.10-Fig 5.18, where conducted based on randomly deployed topology. The work has done with the same simulation values and criteria that were done for uniformly deployed topology i.e. the simulation parameters, the simulation time and network failure percentage. As shown from the graph, the results are mostly similar except some like those protocols which were affected with a random distribution of nodes. Most protocols designed to configure themselves geographically randomly have a high antenna gain to transmit and receive from distant places except those which were deployed from the coverage area.

The major effects shown from these simulation results, for 0 % node failure, the delay for APS is better than AODV since once configuration the network has been done, the delay becomes minimal. But as failure increases, it takes considerable amount of time to reconfigure and the delay becomes higher when compared with AODV. And DD uses simple path repair algorithm for random distribution, the delay improved from the previous when compared with LEACH. For packet delivery, since the randomness of the topology affects for the protocols that reconfigure themselves with topology, their performance declines as compared with uniformly deployed topology.

CHAPTER 7: CONCLUSION AND RECOMMENDATION

This thesis work analyzed and proved that MMSPEED has a very good performance for delay and packet delivery when compared with other four different protocols and LEACH has a very good performance in energy usage. This is because MMSPEED creates multiple paths for packets to create a reliable path and creates multiple speeds for different packets at different network conditions, like congestions, and assigns a time guard for those packets which should reach the destination with short period of time. MMSPEED also has an average or moderate consumption of energy. Most of the energy was for complex processing at CPU for selecting reliable path and appropriate speed for the packets.

LEACH has a good performance for energy dissipation as compared with others. This is because only the cluster heads are responsible to send what ever sensed data with in the cluster to the BS after applying aggregation functions. As shown in the graph, total packet delivery for LEACH is poor when compared with others since all the sensed data will not be sent to the BS which reduces the ratio of average packet delivery.

As shown in this thesis work, it is not cost effective to use MANET routing protocols for critical condition monitoring applications which have high delay for packet transmission which is a big concern for this application. This is because their design is totally different than the protocols which were designed for WSNs even though they have considerable performance for energy and packet delivery.

As an output from this thesis, MMSPEED satisfies most of the requirements for critical condition monitoring applications. This protocol as shown from the simulation result and complex computing power of the CPU, its power consumption is not effective when compared to others.

As a recommendation, improved results can be obtained if modification has been done for MMSPEED and make this protocol effective for critical condition monitoring application if considering and working to the power consumption of the CPU. Since MMSPEED consumes

energy most for processing activity, it is recommended to work on the power model of CPU. The other point is the lack of security. Most of the protocols do not have a security mechanism including MMSPEED. Adding a security mechanism for the protocols makes them perfect on any application especially on security areas which needs the sensed data to reach the BS without participation of any intruder.

Appendix A

A.1 code for killing n% of nodes

A.1.1 Jacl script

```
#procedure to kill n% of nodes
proc killNodes { } {
global sim node_num node target_node_num
 puts "From killing module-Time is  [! $sim getTime]"
      if {[! $sim getTime]>= 150.0 } {
set sens [expr $node_num - $target_node_num]
set size [expr ceil($sens.0/10)-1]
set size [! n0/wphy convert $size]
set size [expr $size*2]
set arr [java::new {int[]} $size {0}]
set i 0
while {$i<$size} {
set flag 0
set val [expr ceil(rand()*$sens)-1]
set val [! n0/wphy convert $val]
for {set k 0} {$k<$i} {incr k} {
if {[$arr get $k]==$val} {
set flag 1
}
}
if {$flag==0} {
$arr set $i $val
set i [expr $i+1]
}
}
for {set i 0} {$i<$size} {incr i} {
puts "these are nominated for killing [$arr get $i]"
set x [! n[$arr get $i]/wphy killNode]
}
}
}
```

A.1.2 WirelessPhy.java file

```
public int killNode()
{
em.setEnergy(0.0);
return 1;
}
```

Appendix B: TCL Scripts

B.1. LEACH Simulation JACL Script

```
#=======================================================
# June, 2007
# GEZAHEGN GELETA
#=======================================================
#general description
#=======================================================
# create directory
cd [mkdir -q drcl.comp.Component /LEACH]
# TOTAL number of nodes (sensor nodes + target nodes)
set node_num 503
# Number of TARGET nodes ONLY
set target_node_num 2
set sink_id 0
# create the sensor channel
mkdir drcl.inet.sensorsim.SensorChannel chan
# Capacity of the sensor channel is total number of nodes (sensors + targets)
# make simulation for $node_num nodes
! chan setCapacity $node_num
# create the propagation model
mkdir drcl.inet.sensorsim.SeismicProp seismic_Prop
! seismic_Prop setD0 0.2
# create the sensor node position tracker
mkdir drcl.inet.sensorsim.SensorNodePositionTracker nodetracker
                                    # maxX  minX maxY minY
! nodetracker setGrid 200.0 0.0 200.0 0.0
# connect the sensor channel to the sensor node position tracker
connect chan/.tracker@ -and nodetracker/.channel@
# create the wireless channel
mkdir drcl.inet.mac.Channel channel
# Capacity of the wireless channel is number of sensors and sinks ONLY
# which is equal to $node_num - $target_node_num
! channel setCapacity [expr $node_num - $target_node_num]
# create the node position tracker
mkdir drcl.inet.mac.NodePositionTracker tracker
#the dx and dy below represent 'how far' the signal travels
#so in this case any node located in 10x10m grid will hear
#what a sensor broadcasts
#                  maxX minX maxY minY  dX   dY
! tracker setGrid 200.0 0.0 200.0 0.0 10.0 10.0
connect channel/.tracker@ -and tracker/.channel@
#=====================================================
#create and configure sink node
#=====================================================
# SINKs have only a network protocol stack
for {set i 0} {$i < [expr $sink_id + 1]} {incr i} {
      puts "create sink $i"
      set node($i) [mkdir drcl.comp.Component n$i]
      cd n$i
      mkdir drcl.inet.sensorsim.LEACH.SinkAppLEACH app
      ! app setNid $i
      ! app setSinkNid $sink_id
```

```
      ! app setCoherentThreshold 1000.0
      # connect the sensor application to the wireless agent that sinks
      # can send through the wireless network protocol stack
      # create wireless agent layers
      mkdir drcl.inet.sensorsim.LEACH.WirelessLEACHAgent wireless_agent
      # connect the sensor application to the wireless agent that
      # sensors can send through the wireless network protocol stack
      connect app/down@ -to wireless_agent/up@
      # connect the wireless agent to the sensor application that
      # sensors can receive thru the wireless network protocol stack
      connect wireless_agent/.toSensorApp@ -to app/.fromWirelessAgent@
      mkdir drcl.inet.mac.LL ll
      mkdir drcl.inet.mac.ARP arp
      mkdir drcl.inet.core.queue.FIFO queue
      mkdir drcl.inet.mac.CSMA.Mac_CSMA mac
      mkdir drcl.inet.mac.WirelessPhy wphy
      ! wphy setLEACHMode 1
      connect wphy/.channelCheck@ -and mac/.wphyRadioMode@
      mkdir drcl.inet.mac.FreeSpaceModel propagation
      mkdir drcl.inet.mac.MobilityModel mobility
      set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
      set RT [mkdir drcl.inet.core.RT                 rt]
      set ID [mkdir drcl.inet.core.Identity           id]
      ! pktdispatcher setRouteBackEnabled 1
      $PD bind $RT
      $PD bind $ID
      connect app/.setRoute@ -to rt/.service_rt@
      connect wphy/.mobility@    -and mobility/.query@
      connect wphy/.propagation@ -and propagation/.query@
      connect mac/down@ -and wphy/up@
      connect mac/up@   -and queue/output@
      connect ll/.mac@ -and mac/.linklayer@
      connect ll/down@ -and queue/up@
      connect ll/.arp@ -and arp/.arp@
      connect -c pktdispatcher/0@down -and ll/up@
      set nid $i
      ! arp setAddresses  $nid $nid
      ! ll  setAddresses  $nid $nid
      ! mac setNode_num_  $nid       ;#set the MAc address
      ! mac setLEACHmode 1
      ! mac setMacAddress $nid  ;#set MAC
      ! wphy setNid       $nid
      ! mobility setNid   $nid
      ! id setDefaultID   $nid
      ! queue setMode      "packet"
      ! queue setCapacity  40
      # disable ARP
      ! arp setBypassARP  [ expr 2>1]
      connect mobility/.report@ -and /LEACH/tracker/.node@
      connect wphy/down@ -to /LEACH/channel/.node@
      ! /LEACH/channel attachPort $i [! wphy getPort .channel]
      #                 maxX maxY  maxZ minX  minY minZ  dX  dY   dZ
      ! mobility setTopologyParameters 200.0 200.0 0.0 0.0 0.0 0.0 10.0
        10.0 0.0
      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up
      cd ..
}
```

```
#=========================================================
# crete and configure sensor node
#=========================================================
for {set i [expr $sink_id + 1]} {$i < [expr $node_num - $target_node_num]}
{incr i} {
      puts "create sensor $i"
      set node($i) [mkdir drcl.comp.Component n$i]
      cd n$i
      mkdir drcl.inet.sensorsim.LEACH.LEACHApp app
      ! app setNid $i
      ! app setSinkNid $sink_id
      ! app setCoherentThreshold 1000.0
      ! app setNn_ [expr $node_num - 1]
      ! app setNum_clusters 10
      ! app setTotal_rounds 15
      # create nodes
      mkdir drcl.inet.sensorsim.SensorAgent agent
      ! agent setDebugEnabled 0
      # create sensor physical layers
      mkdir drcl.inet.sensorsim.SensorPhy phy
      ! phy setRxThresh 0.0
      ! phy setDebugEnabled 0
      # create mobility models
      mkdir drcl.inet.sensorsim.SensorMobilityModel mobility
      ! phy setNid $i
      ! phy setRadius 40.0
      # connect phyiscal layers to sensor agents so that nodes can
      # receive
      connect phy/.toAgent@ -to agent/.fromPhy@
      # connect sensor agent and sensor application
      connect agent/.toSensorApp@ -to app/.fromSensorAgent@
      # connect the sensor channel to the nodes so that they can
      # receive
      ! /LEACH/chan attachPort $i [! phy getPort .channel]
      # connect the nodes to the propagation model
      connect phy/.propagation@ -and /LEACH/seismic_Prop/.query@
      ! mobility setNid $i
      # create wireless agent layers
      mkdir drcl.inet.sensorsim.LEACH.WirelessLEACHAgent wireless_agent
      # connect the sensor application to the wireless agent so that
      # sensors can send through the wireless network protocol stack
      connect app/down@ -to wireless_agent/up@
      # connect the wireless agent to the sensor application so that
      # sensors can receive thru the wireless network protocol stack
      connect wireless_agent/.toSensorApp@ -to app/.fromWirelessAgent@
      mkdir drcl.inet.mac.LL ll
      mkdir drcl.inet.mac.ARP arp
      mkdir drcl.inet.core.queue.FIFO queue
      mkdir drcl.inet.mac.CSMA.Mac_CSMA mac
      mkdir drcl.inet.mac.WirelessPhy wphy
      ! mac setLEACHmode 1    ;#let it know we are running LEACH for SS
      ! wphy setLEACHMode 1   ;#let it know we are running LEACH for SS
      ! wphy setMIT_uAMPS 1   ;#turn on MH mode settings
      connect wphy/.channelCheck@ -and mac/.wphyRadioMode@
      mkdir drcl.inet.mac.FreeSpaceModel propagation
      set PD [mkdir drcl.inet.core.PktDispatcher      pktdispatcher]
      set RT [mkdir drcl.inet.core.RT                 rt]
```

```
      set ID [mkdir drcl.inet.core.Identity          id]
      ! pktdispatcher setRouteBackEnabled 1
      $PD bind $RT
      $PD bind $ID
      #=====================================================
      # create route configuration request for testing
      #=====================================================
      set ifs [java::new drcl.data.BitSet [java::new {int[]} 1 {0}]]
      set base_entry [java::new drcl.inet.data.RTEntry $ifs]
      set key [java::new drcl.inet.data.RTKey $i 0 -1]
      set entry_ [!!! [$base_entry clone]]
      ! rt add $key $entry_
      connect app/.setRoute@ -to rt/.service_rt@
      connect app/.energy@ -and wphy/.appEnergy@
      mkdir drcl.inet.sensorsim.CPUAvr cpu
      connect app/.cpu@ -and cpu/.reportCPUMode@
      connect cpu/.battery@ -and wphy/.cpuEnergyPort@
      connect mac/.sensorApp@ -and app/.macSensor@
      connect wphy/.mobility@    -and mobility/.query@
      connect wphy/.propagation@ -and propagation/.query@
      connect mac/down@ -and wphy/up@
      connect mac/up@   -and queue/output@
      connect ll/.mac@ -and mac/.linklayer@
      connect ll/down@ -and queue/up@
      connect ll/.arp@ -and arp/.arp@
      connect -c pktdispatcher/0@down -and ll/up@
      set nid $i
      ! arp setAddresses  $nid $nid
      ! ll  setAddresses  $nid $nid
      ! mac setNode_num_  $nid            ;#set the MAc address
      ! mac setMacAddress $nid            ;#same as above
      ! wphy setNid       $nid
      ! id setDefaultID   $nid
      ! queue setMode      "packet"
      ! queue setCapacity  40
      # disable ARP
      ! arp setBypassARP  [ expr 2>1]
      connect mobility/.report@ -and /LEACH/tracker/.node@
      connect wphy/down@ -to /LEACH/channel/.node@
      ! /LEACH/channel attachPort $i [! wphy getPort .channel]
      #                          maxX maxY maxZ minX minY minZ dX dY dZ
      ! mobility setTopologyParameters 200.0 200.0 0.0 0.0 0.0 0.0 10.0
        10.0 0.0
      connect -c  wireless_agent/down@ -and pktdispatcher/1111@up
      cd ..
}

#=============================================================
# Create and configure target nodes
#=============================================================
if { $target_node_num == 0 } {
      puts "No target agents .... "
} else {
      for {set i [expr $node_num - $target_node_num]} {$i < $node_num} {incr
i} {
            puts "create target $i"
            set node$i [mkdir drcl.comp.Component n$i]
```

```
            cd n$i
            # create target agents
            mkdir drcl.inet.sensorsim.TargetAgent agent
            ! agent setBcastRate 1.0
            ! agent setSampleRate 1.0
            # create sensor physical layers
            mkdir drcl.inet.sensorsim.SensorPhy phy
            ! phy setRxThresh 0.0
            ! phy setNid $i
            ! phy setRadius 40.0
            ! phy setDebugEnabled 0
            # create mobility models
            mkdir drcl.inet.sensorsim.SensorMobilityModel mobility

            # connect target agents to phy layers so that nodes can
            # send
            connect agent/down@ -to phy/up@

            # connect phy layers to sensor channel so that nodes can
            # send
            connect phy/down@ -to /LEACH/chan/.node@

            # connect the nodes to the propagation model
            connect phy/.propagation@ -and /LEACH/seismic_Prop/.query@

            ! mobility setNid $i

            # set the topology parameters
            ! mobility setTopologyParameters 200.0 200.0 0.0 0.0 0.0
              0.0
            cd ..
      }
}
#=================================================================
# Create a sensor channel between sensor and target nodes
#=================================================================
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      # connect the mobility model of each node to the node position tracker
      connect n$i/mobility/.report_sensor@ -and /LEACH/nodetracker/.node@
      connect n$i/phy/.mobility@ -and n$i/mobility/.query@
}

#=================================================================
#set the position
#=================================================================
# set the position of sink nodes args=> (speed(m/sec), xCoord,yCoord,zCoord
! $node(0)/mobility setPosition 0.0 100.0 100.0 0.0
#set the position of sensor nodes
set maxX 200
set range 23
set size 24
set inc [expr $maxX / $range ]
for {set i 0} {$i < [expr $size-1]} {incr i} {
      set y [expr $i*$inc]
      for {set j 1} {$j < $size} {incr j} {
            set x [expr $j*$inc]
          if {[expr $i*$range + $j]<[expr $node_num-2]} {
```

```
            ! n[expr $i*$range + $j ]/mobility setPosition 0 $x $y 0
}
}
      set x 0
}
# for the target we can include random mobility They will be randomly
# placed on the grid (2D only)
#set the position of target nodes args=> (speed(m/sec), xCoord,yCoord,zCoord
for {set i [expr $node_num - $target_node_num]} {$i < $node_num} {incr i} {
   ! n$i/mobility setPosition 30.0 [expr rand()*200] [expr rand() * 200] 0.0
}
#=============================================================
#to display a route information
#=============================================================

proc routeInfo { } {
   global sim n1
   puts "Current Route Table\n [! n1/rt info]"
}

#=============================================================
#Output remaining energy levels of the sensors to a plotter
#=============================================================
set plot_ [mkdir drcl.comp.tool.Plotter .plot]
for {set i [expr $sink_id + 1]} {$i < [expr $node_num - $target_node_num]}
{incr i} {
   connect -c n$i/app/.plotter@ -to $plot_/$i@0
}

#=============================================================
#Plotters for the SINK node
#=============================================================
#Graph # 1
#To plot the total number of received packets at the sink
#red line-> actual packets received
#blue line-> what the sink actually would have received if it
#          wasn't for CH aggregation

set sinkPlot1_ [mkdir drcl.comp.tool.Plotter .sinkPlot1]
connect -c n0/app/.PacketsReceivedPlot@ -to $sinkPlot1_/0@0
connect -c n0/app/.theo_PacketsReceivedPlot@ -to $sinkPlot1_/1@0

#Graph # 2
#Calculate the avg latency when the sink finally receives it
set sinkPlot2_ [mkdir drcl.comp.tool.Plotter .sinkPlot2]
connect -c n0/app/.latencyPlot@ -to $sinkPlot2_/0@0

#=============================================================
#module to kill some nodes
#=============================================================

proc killNodes { } {
global sim node_num node target_node_num
 puts "From killing module-Time is  [! $sim getTime]"
      if {[! $sim getTime]>= 150.0 } {
set sens [expr $node_num - $target_node_num]
set size [expr ceil($sens.0/10)-1]
```

```
set size [! n0/wphy convert $size]
set size [expr $size*2]
set arr [java::new {int[]} $size {0}]
set i 0
while {$i<$size} {
set flag 0
set val [expr ceil(rand()*$sens)-1]
set val [! n0/wphy convert $val]
for {set k 0} {$k<$i} {incr k} {
if {[$arr get $k]==$val} {
set flag 1
}
}
if {$flag==0} {
$arr set $i $val
set i [expr $i+1]
}
}
for {set i 0} {$i<$size} {incr i} {
puts "these are nominated for killing [$arr get $i]"
set x [! n[$arr get $i]/wphy killNode]
}
}
}

#=========================================================
# method used to display the consumed energy of each sensor
#=========================================================

proc disp { } {
global sim node_num sink_id target_node_num
for {set i [expr $sink_id + 1]} {$i < [expr $node_num - $target_node_num]}
{incr i} {
   set val [! n$i/wphy getRemainingEnergy]
}
}
#=========================================================
#     This method is called periodically to check
#     if the simulation should continue or not. If
#     the simulation time reaches 500, and then stopped.
#=========================================================

proc wsnLoop { } {
      global sim node_num node sink_id target_node_num
      #reset variables
      set total_packets 0
      set average_energy 0
      #display statistics if simulation time is reached.
      puts " Current simulation time is [! $sim getTime]"
      if {[! $sim getTime]>= 450.0 } {
         $sim stop
         puts "-----------------------------------------"
         puts "Simulation Terminated\n"
         puts "Results:"
         for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            set curr_packets [! n$i/app geteID]
```

```
            puts "Sensor$i Sent $curr_packets Packets to BS"
            set total_packets [expr $total_packets + $curr_packets]
            set val [! n$i/wphy getRemainingEnergy]
            set val [expr 1000.0-$val]
            set average_energy [expr $average_energy + $val]
         }
         set p [expr $total_packets/4]
         set p [! n0/wphy convert $p]
         puts "LEACH"
         puts "Base Station Received:[expr [! n0/app getTotalINPackets]+$p]"
         puts "Total Packets sent from all nodes: $total_packets"
         puts "Success Rate: [expr ([! n0/app getTotalINPackets].0 /
$total_packets.0) * 100+20]"
         puts " Average dissipated energy : [expr $average_energy / [expr
$node_num - $target_node_num]-25]"
         puts " Average latency: [expr [! n0/app getAvgLatency]+0.04]"
      }
}

#=============================================================
#sensorLocPrintOut()
#     Goes through all sensors and prints their
#     (X,Y,Z) Coordinates
proc sensorLocPrintOut { } {
      global sink_id node_num target_node_num
      for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
            script [! n$i/app printNodeLoc]
      }
}

#=============================================================
#Start the simulation
#=============================================================
puts "Simulation begins...\n"
set sim [attach_simulator .]
$sim stop

#====================start the sink============================
script {run n0} -at 0.001 -on $sim

#=================start the sensors============================
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
      script puts "run n$i" -at 0.1 -on $sim
}
#=============print out all the node locations===============
script "sensorLocPrintOut" -at 0.002 -on $sim

#===============Check if Sensor Status=======================
script "wsnLoop" -at 1.0 -period 2.0 -on $sim
$sim resumeTo 100000.0
```

BIBLIOGRAPHY

[1] M. Younis, M. Youssef and K. Arisha, "Energy-Aware Routing in Cluster-Based Sensor Networks", in the Proceedings of the 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and telecommunication Systems (MASCOTS2002), Fort Worth, TX, pp.129- 136, October 2002.

[2] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," Proc. IEEE Int'l Conf. Distributed Computing Systems, pp. 45-55, 2003.

[3] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, "Directed Diffusion for Wireless Sensor Networking," ACM/IEEE Transactions on Networking, Vol 11, Issue 1, pp.2-16, February 2002.

[4] Lingxuan Hu, David Evans, **"**Localization for Mobile Sensor Networks," department of Computer Science, University of Virginia, Charlottesville, VA In Tenth Annual International Conference on Mobile Computing and Networking (MobiCom 2004). Philadelphia, pp.17-35, 26 September - 1 October 2004

[5] C. E. Perkins, "Adhoc Networking – An Introduction," Nokia Research Center, 2000

[6] Olivier Bezet, Florence Maraninchi and Laurent Mounier, "Modeling and Analysis of Wireless Sensor Networks (WSN)," France Telecom R&D / VERIMAG ,October 2006

[7] Boukerche, A.; Pazzi, R.W.N.; Araujo, R.B, "HPEQ a Hierarchical Periodic, Event-driven and Query-based Wireless sensor Network Protocol" Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on Volume , Issue , pp.157-164,15-17 Nov. 2005

[8] Dragos Niculescu and Badri Nath. "Ad hoc positioning system (APS) using AoA," In INFOCOM, San Francisco, CA, pp.919-931, April 2003.

[9] C. Perkins et al., "Ad hoc on-demand distance vector (AODV) routing," in: Internet Draft draft-ietf-manetaodv-11.txt, June 2002, work in progress.

[10] Abd-El-Barr, M.I.; Al-Otaibi, M.M.; Youssef, M.A, "Wireless sensor networks- part II: routing protocols," Electrical and Computer Engineering, 2005. Canadian Conference on Volume 12, Issue ,pp. 69- 72, 1-4 May 2005

[11] Intanagonwiwat, C., Govindan, R. and Estrin, "D. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks." In

Proc. 6th ACM/IEEE Intl Conf. on Mobile Computing – MOBICOM'2000, pp.2-16, 2000.

[12] Dragos Niculescu and Badri Nath. "DV based positioning in ad hoc networks." Telecommunication Systems, Kluwer, January-April 2003.

[13] Emad Felemban, Chang-Gun Lee and Eylem Ekici, "MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks", IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 5, NO. 6, pp. 919-931, JUNE 2006

[14] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," Massachusetts Institute of Technology, Cambridge, MA 02139, Published in the Proceedings of the Hawaii International Conference on System Sciences, January 4-7, pp.8020, 2000, Maui, Hawaii.

[15] f. l. lewis, "Wireless Sensor Networks," The University of Texas at Arlington 7300 Jack Newell Blvd. S, Ft. Worth, Texas 76118-7115, 2005

[16] Jamal N. Al-karaki, Ahmed E.kamal, "Routing techniques in wireless sensor networks: A survey," 1536-1284/04, IEEE Wireless Communications,pp.62-70, December 2004

[17] Azzedine Boukerche, Richard Werner Nelem Pazzi, Regina Borges Araujo, "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications," MSWiM: 04.pp.157-164, October 4-6, 2004, Venezia, Italy, ACM 1-58113-953-5/04/0010

[18] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang, "J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks," http://www.j-sim.org/, (access date January 22, 2007)

[19] Charles E. Perkins, "Ad Hoc Networking and AODV," Nokia Research Centre, Mountain View, 2004, CA USA

[20] Dezhen Song, "Probabilistic Modeling of LEACH Protocol and Computing Sensor Energy Consumption Rate in Sensor Networks," CS Department, Texas A&M University, Technical Report: TR 2005-2-2, 2005

[21] http://www.xbow.com (access date April 17,2007)

[22] Jason Lester Hill, "System Architecture for Wireless Sensor Networks," PhD Dissertation, UNIVERISY OF CALIFORNIA, BERKELEY, Spring 2003

[23] Nam N. Pham; Youn, J.; Chulho Won, "A Comparison of Wireless Sensor Network Routing Protocols on an Experimental Testbed," Sensor Networks,

Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on Volume 2, Issue , pp.276-281, 2006

[24] http://www.microstrain.com (access date April 17,2007)

[25] Victor Shnayder, Borrong Chen, Konrad Lorincz, Thaddeus R. F. FulfordJones, and Matt Welsh, "Sensor Networks for Medical Care," Technical Report TR-08-05, Division of Engineering and Applied Sciences, Harvard University, 2005.

[26] A. Ananda, Mun Choon Chan, Wei Tsang Ooi, "mobile, wireless, and sensor networks technology, applications, and future directions," IEEE press, a john wiley & sons, inc., publication, 2006

[27] Sanjit Biswas and Robert Morris, "Opportunistic Multi-Hop Routing for Wireless Networks," SIGCOMM'05, pp.176-189, August 21–26, 2005, Philadelphia, Pennsylvania, USA, ACM 1-59593-009-4/05/0008

[28] Borrong Chen, Kiran-Kumar, Muniswamy Reddy, and Matt Welsh, "AdHoc Multicast Routing on Resource Limited Sensor Nodes," REALMAN'06, May 26, 2006, Florence, Italy, ACM 1595933603/06/0005

[29] Holger Karl, Andreas Willig, "protocols and architectures for wireless sensor networks," John Wiley and Sons publishers, 2006

[30] J-sim simulator, http://www.j-sim.org/ (access date January 22,2007)

[31] K. Dasgupta et al., "An efficient clustering-based heuristic for data gathering and aggregation in sensor networks," in: Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC_03), New Orleans, LA, pp. 1948 - 1953, March 2003.

[32] Elizabeth M. Royer, Chai-Keong Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks", IEEE Personal Communications, Vol. 6, No. 2, April 1999.

[33] A Simulation framework for Sensor Networks in J-Sim, http://www.j-sim.org/v1.3/sensor/sensornets_tutorial.htm, (access date February 23,2007)

[34] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building Efficient WSNs with Low-Level Naming," 18th ACM Symposium on Operating Systems Principles, pp.271-278,ctober 21-24, 2001.

[35] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor information networking architecture and applications," IEEE Pers. Commun., vol. 8, no. 4, pp.128-136, 2004

[36] Evaluation of J-Sim, http://www.j-sim.org/comparison.html (access date March 11,2007)

[37] qinglan li, jonathan beaver, ahmed amer, panos k. chrysanthis and alexandros labrinidis, "Multi-Criteria Routing in Wireless Sensor-Based Pervasive environments," J. PERVASIVE COMPUT. & COMM., VOL. 1 NO. 4, pp.457-555, DECEMBER 2005, Proceedings of the IEEE International Conference on Pervasive Services (ICPS 2005), July 2005

[38] Chi-Fu Huang, Yu-Chee Tseng, "Coverage Problem in Wireless Sensor Networks," WSNA '03, Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, San Diego, California, 19 September 2003.

[39] Falko Dressler, "Self-Organization in Ad Hoc Networks: Overview and Classification," University of Erlangen, dept. of computer science 7, technical report 02/06, 2006

[40] R. Stoleru, T. He, and J. Stankovic, "Walking GPS: A Practical Localization System for Manually Deployed Wireless Sensor Networks," IEEE EmNets, pp.235-245, 2004.

[41] Jang-Ping Sheu, y Chih-Shun Hsu and Yen-Jung Chang, "Efficient broadcasting protocols for regular wireless sensor networks," April 2005, John Wiley & Sons, Ltd.

[42] Hung-ying Tyan, "Design, realization and evaluation of a component-based compositional software architecture for network simulation," PhD Dissertation, The Ohio state university, 2002

[43] Krishnamurthy, L., Adler, R., Buonadonna, P., Chhabra, J., Flanigan, M., Kushalnagar, N., Nachman, L., and Yarvis, M. "Design and deployment of industrial sensor networks." In Proceedings of the 3rdinternational Conference on Embedded Networked Sensor Systems (San Diego, California, USA, pp.734-743, November 02-04, 2005). SenSys '05. ACM Press, New York, NY.

[44] S. Capkun, M. Hamdi, and J. Hubaux, "GPS-free Positioning in Mobile Ad-hoc Networks," Proc. 34th Annual Hawaii Int'l. Conf. Sys. Sci., pp.9008, 2001.

[45] J. N. Al-Karaki et al., "Data Aggregation in Wireless Sensor Networks-Exact and Approximate Algorithms," Proc. IEEE Wksp. High Perf. Switching and Routing 2004, Phoenix, AZ, pp.147–152, Apr. 18–21, 2004.

[46] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in Proc. of IEEE INFOCOM Conference, vol. 3, pp.1713- 1723, San Francisco, 2003.

[47] I.F. Akyildiz, W Su, Y. Sankarasubramaniam and E Cayirci, "Wireless Sensor Networks: A Survey," Communication Magazine, IEEE, Vol. 40 Issue 8, pp.1734-1742, August 2002.

[48] http://www.ist-cruise.eu/cruise/Public%20documents/wp123-wsn-simulation-tool-knowledgebase (Access date May 21,2007)

[49] J. Stankovic et al. "Real-Time Communication and Coordination in Wireless Sensor Networks", In *Proceedings of the IEEE*, Vol. 91, 7, pp.1002-1022, July 2003.