2005

Simulation study for wireless sensor networks and load sharing routing protocol to increase network life and connectivity

Ankur Suri
Louisiana State University and Agricultural and Mechanical College

SIMULATION STUDY FOR WIRELESS SENSOR NETWORKS AND LOAD SHARING ROUTING PROTOCOL TO INCREASE NETWORK LIFE AND CONNECTIVITY

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Systems Science

in

The Department of Computer Science

By

Ankur Suri
B.Sc(Honors) Delhi University, New Delhi, 1997
M.Sc Mathematics Indian Institute of Technology, New Delhi, 1999
December 2005

Acknowledgements

I would like to express my sincere gratitude to Dr. S. Sitharama Iyengar, my Chair, for his invaluable guidance and encouragement extended throughout the study. His tenacious supervision, helpful suggestions, patience and time deserve a special mention.

I would also like to thank my Co-Chair Dr Anitra Wilson for her invaluable support and encouragement. Her guidance and suggestions deserve a special mention.

I would like to express my appreciation to my committee member Dr. Rajgopal Kannan for his support and suggestions.

I would also like to thank my colleagues Cariappa Mallanda, Vatsalya Kunchakara, Neelay Shah for their suggestions on implementation of this thesis.

Last, but not the least, I would like to gratefully acknowledge Department of Computer Science, Louisiana State University for providing the resources during the project.

Table of Contents

Acknowledgements……..**ii**

Abstract.....**v**

Chapter 1 : Introduction.....**1**

Chapter 2 : Wireless Sensor Nodes.....**5**

Chapter 3 : Sensor Simulators.....**7**
3.1 Currently Available Simulators.....9
3.2 OMNeT++.....10

Chapter 4 : LSU-SensorSimulator.....**12**
4.1 High Level Design.....12

Chapter 5 : Design Approach.....**16**
5.1 Target Node.....16
5.2 Sensor Channel.....16
5.3 Sensor Node.....18
5.4 Coordinator Module.....18
5.5 Hardware Modeling.....19
5.5.1 Battery Module.....19
5.5.2 CPU Model.....19
5.5.3 Radio Model.....20
5.6 Software Model.....20
5.6.1 Wireless Channel.....21

Chapter 6 : Implementation Details for LSU-SensorSimulator.....**23**
6.1 Directed Diffusion with GEAR Implementation.....23
6.2 MAC 802.11.....25

Chapter 7 : Experimental Setup for LSU-SensorSimulator.....**29**

Chapter 8 : Results for LSU-SensorSimulator.....**34**

Chapter 9 : Buddy Load Sharing Routing Protocol.....**37**

Chapter 10 : Algorithm for Buddy Load Sharing Routing Protocol.....**42**
10.1 One Hop Buddy Load Sharing Algorithm.....42
10.2 N Hop Buddy Load Sharing Algorithm.....43

Chapter 11 : Experimental Setup for Buddy Load Sharing Routing Protocol.....**46**
11.1 Directed Diffusion.....46

11.2 GEAR.....47

Chapter 12 : Results.....49

Chapter 13 : Conclusion and Future work 51

References.....52

Vita.....54

Abstract

LSU SensorSimulator is a framework for simulating wireless sensor networks. It is a customizable and extendible simulator, which allows testing and analyzing software for wireless sensor networks. The users can subclass the framework classes and customize the behavior of various network layers. This subclassing gives a way to the developers an opportunity to analyze and investigate, phenomenological, networking, robustness and scaling issues, to explore arbitrary algorithms for distributed sensors, independent of hardware constraint. The results are compared against the simulation results for ns-2 for routing protocols Directed Diffusion and GEAR. Through the comparison of results for scalability, performance and memory utilization it is observed that LSU SensorSimulator performs much better. Buddy load sharing routing protocol is a routing protocol which can be combined with any geographically aware routing protocol to increase the network life and connectivity. The performance of Buddy load sharing algorithm for network life, and it is found that for a very negligible overhead the network life and connectivity and be improved by buddy load sharing.

Chapter 1 : Introduction

Wireless Sensor Networks (WSN) comprise of numerous tiny sensors that are deployed in spatially distributed terrain. WSN are based on the concept of proactive computing. With the proactive computing model, computers will anticipate our needs and sometimes take action on our behalf. Sensor networks and proactive computing can help us improve productivity, have data from places which are otherwise inaccessible or to costly to monitor [13].

Sensor networks were first proposed by researchers at the Defense Advanced Research Projects Agency (DARPA) [14]. These sensors were to be used for various purposes like to detect poisonous gas or to detect tanks etc. For such like purposes it was desired that the sensor nodes be of very small size. These sensor nodes can form a network and data collected from one sensor node can be transferred to another node. This transfer of data from one node to another continues until the data reaches the final destination.

Moore's Law predicts that with the technological advancements the number of transistors on a microchips will double every two years, and for this reason microprocessors with a given processing power are becoming smaller and cheaper with each passing day. Microelectromechanical systems called MEMS enable the production of velocity sensors, thermometers and very tiny low-power radio components and are extremely inexpensive. Wireless sensor nodes are made of three parts: microprocessors, sensors and low powered radios. The wireless sensor nodes made by UC Berkeley for Smart Dust project were nicknamed “motes” [15].

There are many technical shortcomings which need to be overcome before WSN can be practically used. Due to the small size, the nodes are very constrained in all the resources. They have limited processing speed, storage capacity and bandwidth. Because of the small battery size, the life time of a node is dependent on its capacity to conserve power. All these constraints make it impossible to use the same software design, hardware design and network architecture as in desktops as desktops do not have such limited resources. Thus hardware design, software design and network architecture need to be redesigned to meet the special needs of sensor nodes.

Due to small size these sensor nodes can be deployed in ways that wired sensor systems couldn't be deployed. This feature of WSN has opened new ways for scientists and engineers to observe physical phenomena. These WSN are made of a large number of nodes, which are self contained, battery powered computers with very small computation power and battery life. These nodes can measure light, temperature, humidity and other environmental attributes.

These wireless sensor nodes have the capability to form a network and collect data from their immediate environment and transmit it. These sensor nodes have various applications, such as sensors buried in the soil can take measurements from the soil and manage irrigation and fertilizer use. For example if there is presence of some fungus in the soil, the sensors on detecting this can activate pesticides to prevent further damage. Similarly sensors can be used to monitor the pollution level in a river. Sensor nodes with vibration and temperature monitors can be used in manufacturing plants so as to reduce

equipment downtime. These nodes on machines sense any change in machine vibration and temperature and send out warning messages indicating that machine is behaving in an unexpected way. Some of the places where WSN have been successfully used are [13]

- British Petroleum(BP) one of the world's largest petroleum and petrochemicals company, is collaborating with Intel to use WSN to provide continuous vibration monitoring of the engines of the oil tankers used by BP in Shetland Islands in northern Scotland.
- A robust sensor network on Great Duck Island, off the coast of Maine aids biologists in the study of Leach's storm petrels, a specie of seabird that have mysteriously selected this location as their breeding ground. (Intel Research/UC Berkeley).
- As part of the DARPA NEST program, researchers demonstrated a sensor network at MacDill Air Force Base that can detect, classify, and track soldiers and vehicles in difficult-to-monitor open spaces such as desert battlefields. (Ohio State University)
- A sensor network deployed in an Oregon vineyard guides irrigation and planting, increasing crop yield. (Intel Research/ King Family Farms/AgCanada)
- Inside an experimental smart home at Intel’s Oregon campus, a sensor network is under development that could someday keep tabs on an Alzheimer’s patient’s vital signs while reminding him how to warm up his lunch. (Intel Research) On the San Andreas Fault, a network of motes

equipped with seismometers calculate the depth of the fault, locate accumulating stress, and may eventually improve earthquake prediction. (UCLA Department of Earth and Space Sciences/ Center for Embedded Networked Sensing)

- Motes mounted on the treetops of UC Botanical Garden's Mather Redwood Grove sample environmental data in a cross section of the canopy to help scientists understand the massive plants' physiology. (UC Berkeley/Intel Research)
- Motes that measure vibration signatures on manufacturing equipment are being tested for "pre-emptive maintenance applications" to reduce downtime in semiconductor fabrication facilities. (Intel Research/Intel Technology and Manufacturing Group)

This thesis work has two parts. The first part explains the design, implementation of LSU-SensorSimulator and compares it with ns-2. In the second part a new routing protocol "buddy load sharing protocol " is proposed to increase the network life and increase network connectivity. The buddy load sharing protocol is implemented on LSU-SensorSimulator with GEAR and Directed Diffusion and comparative results are discussed.

Chapter 2 : Wireless Sensor Nodes

Wireless Sensor nodes consist of three basic components

- microprocessor,
- sensor
- low powered radio.

Figure 1* Wireless Sensor Nodes

- **(adopted from path.berkeley.edu smart dust project)**

As sensors can be programmed to transmit only relevant data, the usage of motes in real life applications is tremendous. While sensors have been present in commercial products such as automobiles, what sets motes apart is their ability to network using radio frequencies. Sensor nodes are embedded devices which combine sensing, communication and computation. Since the nodes have very small processing power they have to use smaller versions of operating

systems. One of the widely used operating system is TinyOS16]. TinyOS is an open-source operating system designed for wireless embedded sensor networks. It uses a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. The programming language of TinyOS is stylized C that uses a custom compiler 'NesC'. TinyOS was initially developed by the U.C. Berkeley EECS Department.[a]. TinyOS provides interfaces for networking, scheduling and other components interface. TinyOS is an operating system on which various algorithms used to govern various activities of the nodes are implemented. If the algorithm implemented in the nodes needs to be changed that can be done by simply telling one node about the change and this node will pass on the instructions to the other nodes.

Sensor nodes are event driven. The nodes are asleep most of the time. The processors are activated only when the nodes receive a message or when the node acquires new data. NesC supports the motes' reactivity to their environment. The component model simplifies the creation of applications and aggregation of data.

Chapter 3 : Sensor Simulators

Because of the constraints imposed on sensor networks such as energy limitations, decentralized collaboration, fault tolerance etc algorithms for sensor networks are complex. In traditional networks, to analyze performance the techniques used are analytical methods, computer simulation, and physical measurements. The analytical techniques which have been quite effective in traditional networks fail in sensor networks because of the complexity of the algorithms. Because of the high cost of deploying large-scale WSN's and many unsolved research problems there are very few sensor networks in existence, so physical measurements is not possible. Computer simulation comes in as a very reliable resource for analyzing the performance in a very realistic manner.

One of the most widely used simulators for traditional networks is ns2[4]. In a recent report [17] the following paragraph summarizes the need for a new simulator.

"ns2, perhaps the most widely used network simulator, has been extended to include some basic facilities to simulate Sensor Networks. However, one of the problems of ns2 is its object-oriented design that introduces much unnecessary interdependency between modules. Such interdependency sometimes makes the addition of new protocol models extremely difficult, only mastered by those who have intimate familiarity with the simulator. Being difficult to extend is not a major problem for simulators targeted at traditional networks, for there the set of popular protocols is relatively small. For example, Ethernet is widely used for wired LAN, IEEE 802.11 for wireless LAN, TCP for reliable transmission over

unreliable media. For sensor networks, however, the situation is quite different. There are no such dominant protocols or algorithms and there will unlikely be any, because a sensor network is often tailored for a particular application with specific features, and it is unlikely that a single algorithm can always be the optimal one under various circumstances.

Many other publicly available network simulators, such as JavaSim, SSFNet, Glomosim and its descendant Qualnet, attempted to address problems that were left unsolved by ns2. Among them, JavaSim developers realized the drawback of object-oriented design and tried to attack this problem by building component-oriented architecture. However, they chose Java as the simulation language, inevitably sacrificing the efficiency of the simulation. SSFNet and Glomosim designers were more concerned about parallel simulation, with the latter more focused on wireless networks. They are not superior to ns2 in terms of design and extensibility."

The design of wireless sensor networks requires us to simultaneously consider the effects of several factors such as energy efficiency, fault tolerance, quality of service demands, synchronization, scheduling strategies, system topology, communication and coordination protocols. The following sections describe the structural design of a new simulator for wireless sensor networks that is based on the discrete event simulation[12] framework OMNeT++ and results that demonstrate that the new simulator executes at least an order of magnitude faster than ns2 while using memory more efficiently. The design proposed is general, but for results comparison with ns2 we have implemented

IEEE 802.11 MAC layer and Directed Diffusion[1][2] integrated with the Geographical and Energy Aware Routing (GEAR)[22] protocol.

3.1 Currently Available Simulators

- ns2 is a well-established discrete event simulator that provides extensive support for simulating TCP/IP, routing and multicast protocols over wired and wireless networks [4]. Radio propagation model based on two ray ground reflection approximation and a shared media model in the physical layer, an IEEE 802.11 MAC protocol in the link layer and an implementation of dynamic source routing for the network layer were developed in the Monarch project [8].
- SensorSim builds on ns2 and claims to include models for energy and the sensor channel [5][14]. At each node, energy consumers are said to operate in multiple modes and consume different amounts of energy in each mode. The sensor channel models the dynamic inter-action between the physical environment and the sensor nodes. This simulator is no longer being developed and is not available.
- OPNET Modeler is a commercial platform for simulating communication networks [23]. Conceptually, OPNET model comprises processes that are based on finite state machines and these processes communicate as specified in the top-level model. The wireless model is based on a pipelined architecture to determine connectivity and propagation among nodes. Users can specify frequency, bandwidth, and power among other characteristics including antenna gain patterns and terrain models.

- J-Sim is another object-oriented, component-based, discrete event, network simulation framework written in Java [18]. Modules can be added and deleted in a plug-and-play manner and J-Sim is useful both for network simulation and emulation by incorporating one or more real sensor devices. This framework provides support for target, sensor and sink nodes, sensor channels and wireless communication channels, physical media such as seismic channels, power models and energy models.
- GlomoSim is a collection of library modules, each of which simulated a specific wireless communication protocol in the protocol stack [20]. It is used to simulate Ad-hoc and Mobile wireless networks.

3.2 OMNeT++

OMNeT++ [12] Objective Modular Network Test-bed in C++ is a public-source, component-based, modular and open-architecture simulation environment with strong GUI support and an embeddable simulation kernel. Its primary application area is the simulation of communication networks, but because of its generic and flexible architecture, it has been successfully used in other areas. The main features of OMNeT++ are:

- Discrete event simulator
- Message driven
- Programming languages used are C++ and Tcl/Tk.
- Thread/co-routine based programming and finite state machine(FSM) model are supported.

- Allows hierarchically nested modules with no limit on the depth. This allows the user to reflect the logical structure of the actual model.
- Modules can modify their behavior based on module parameters. These parameters are also used as shared variables between modules.
- Modules at the lowest level of the module hierarchy are to be provided by the user, and they include the algorithms in the model.
- Provides support for parallel execution.
- Has different user interfaces for different purposes: debugging, demonstration and batch execution. Also provides support for recording data vectors and scalars in output files.
- Provides well-documented API for simulation modeling.
- Simulation runs are easy to configure and run using initialization files.
- Several random number generators for different distributions are provided.

The simulated objects such as modules, gates, connections, etc are either statically created at the beginning of the simulation using the configuration file or dynamically during simulation.

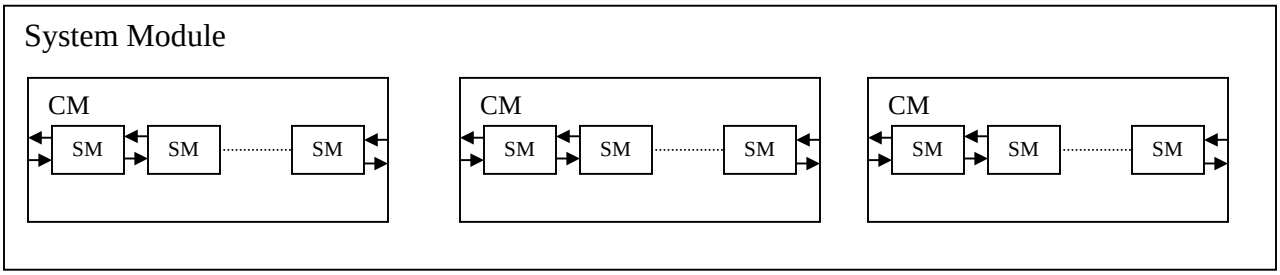


Figure 2 System Module

Chapter 4 : LSU-SensorSimulator

4.1 High Level Design

The LSU-SensorSimulator is a framework to model and simulate a sensor network scenario. The figure 3 illustrates sensor node model representing the network stack and sensor applications. The power model represents the hardware of the sensor node consisting of CPU, sensor and RF trans-receiver. The two models act in parallel to simulate the hardware and software. The hardware model updates its state based on the function carried out by the sensor node model. The power model has a single finite energy source and multiple consumers. The consumers are radio, CPU and other sensing devices as illustrated in figure 4. The consumers triggered by their activities report their power state changes to battery, and thus the remaining energy is updated.

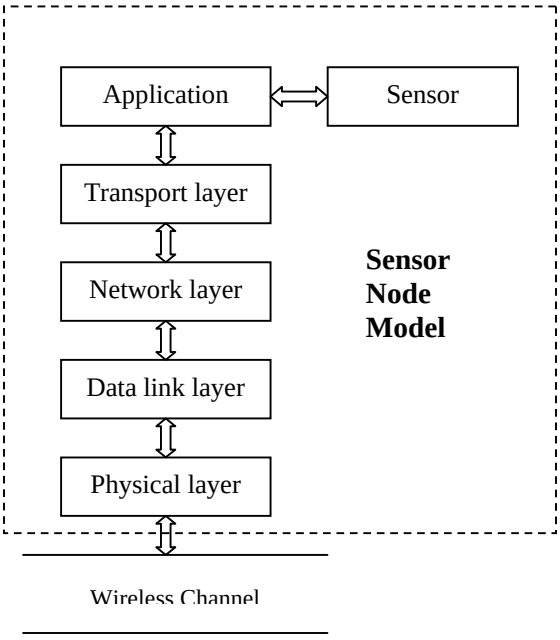


Figure 3 Sensor Node Model

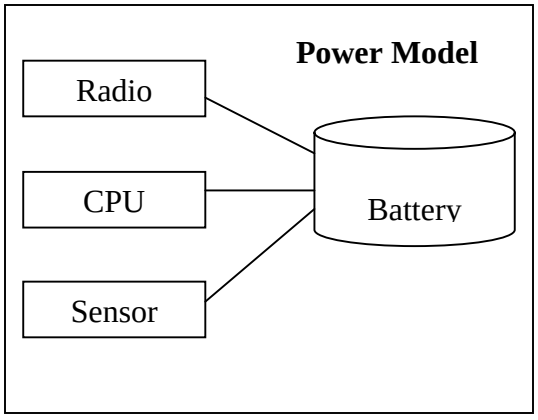


Figure 4 Power Model

LSU-SensorSimulator framework consists of the network of sensor nodes that can communicate by wireless means. The layers of network stack in the sensor model are configurable based on the protocol needed for the simulation. Users can write there own code and integrate it into the framework. The simulation and network parameters are set in the configuration file(omnetpp.ini), thus the parameters can be changed without any changes in the code. The broad high level illustration of any sensor network is shown in figure 5.

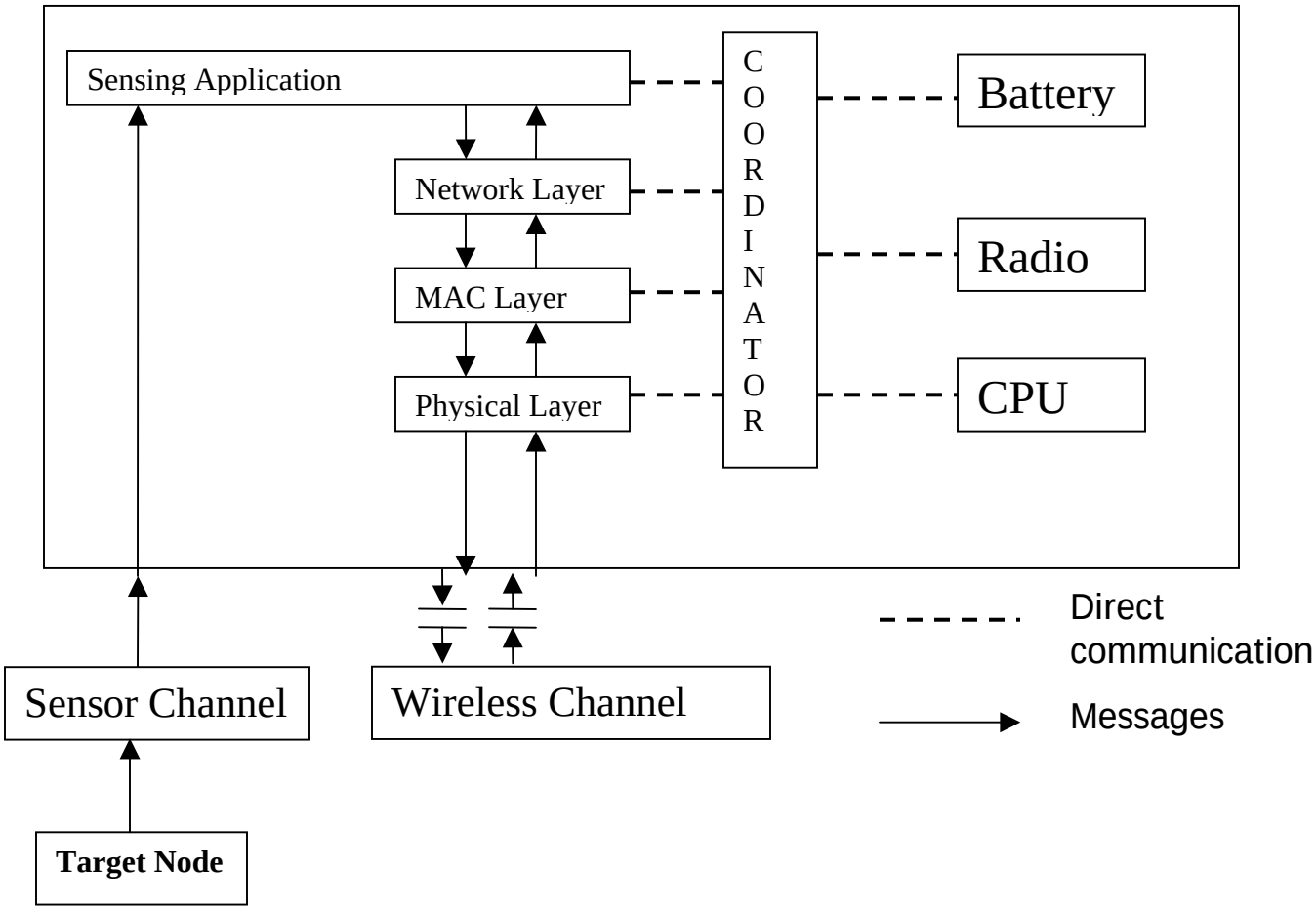


Figure 5 High Level Design for Sensor Network

The sensor simulation framework can be described as the sensors detect events that are generated by a target or object in the environment near the

sensors. The event or observation(data collected) is reported back to the base station using multi-hop routing protocol. The sensor node detects or senses events in its environment, the Target Node is a node that generates the events and the Sink is the base station that consumes the data or the node that sends out query to the network. The events sensed by the sensor channel are propagated across the network through the wireless channel. In the wireless channel different propagation models to prorogate data in the wireless medium are implemented.

Sensor node uses the network protocol stack to detect the events in its environment (generated by the target node) and it sends out messages to other nodes in the network based on the different protocols implemented at each layer of the protocol stack. The functioning of the framework and abstract view of sensor network as shown in figure 5 is described below:

The target node moves across the network at a configurable speed. The target node sends stimuli to the sensor channel. The sensor channel in turn will pass on the stimuli to only those sensor nodes in the vicinity of the target node. A sensor node is able to receive the stimuli only if the signal strength power of the received packet is above a certain threshold. The propagation model configured at the sensor channel determines the attenuation of the signal and the received signal strength power.

Various algorithms or protocols for data aggregation, clustering, security and other in-network processing are implemented in the sensor node at the

Sensing Application Layer. The data collected by the node has to be sent to the base station through the wireless channel. A sensor node can transmit data for a

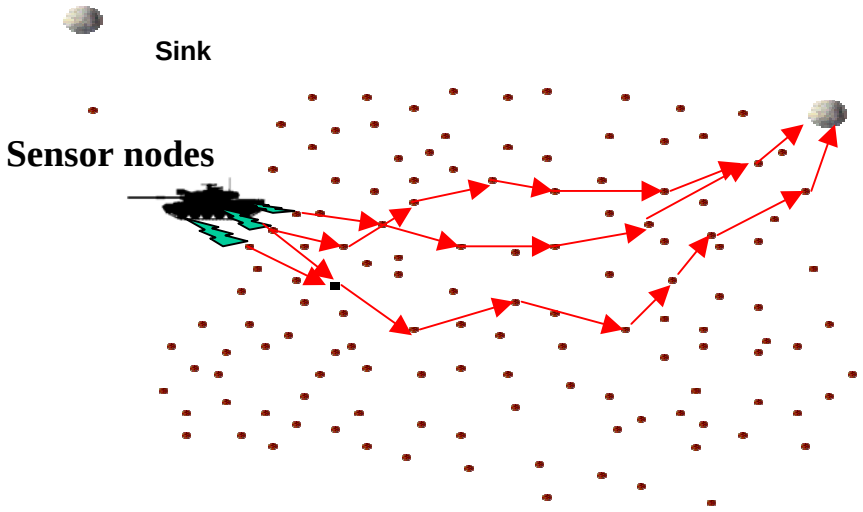


Figure 6 Sensor Network

distance of around 30 feet, thus multi-hop route needs to be taken to send data to the base station. The other sensor nodes act as routers which just help in passing the data from the sensing node to the base station. Since the sensor nodes have finite amount of energy and are mostly in hostile conditions they might die at any point of the network life due to energy depletion or environmental conditions. This would lead to a change in the network topology, thus the routing protocol should be able to take care of the dynamic changes in the topology. Also the routing protocol should be able to transmit data to the base node in a timely, reliable and energy efficient manner. In the frame work two routing protocols have been implemented: directed diffusion and the Geographic aware routing protocol (GEAR) to test the functioning of the simulation framework and to show the proof of concept of a sensor network protocol implementation. The results have been compared with ns2 results for the same two protocols.

Chapter 5 : Design Approach

The simulator has a layered design. The different layers and modules communicate with each other by passing messages. The general architecture of a sensor network is as shown in figure 5. SensorNetwork module contains different modules like TargetNode, SensorNode, WirelessChannel and SensorChannel.

5.1 Target Node

TargetBase class is the base class that represents the Target Node. The TargetBase has the base class functionalities that are essential for any TargetNode such as the position of the target node and the ID. TargetNodeSimple extends the TargetBase and has the functional implementation of the TargetNode. The TargetNode module maintains Gate connection with the sensor channel. The TargetNodeSimple class generates stimuli and passes the message to the sensor channel. The mobility model provides the functionality of the TargetNode movement thereby generating stimuli at various points in the network.

5.2 Sensor Channel

The SensorChannel Module and the SensorChannel Base class represent the Sensor channel. The SensorChannel module maintains Gate connections to SensorNode Module as well as to the TargetNode. The SensorChannel Base is an abstract class for SensorChannel property classes

The location information of all the sensor nodes is maintained at the Network level, which is the parent module of the Sensor Channel. This kind of an abstraction has been designed, as the network module that encompasses the whole simulation model must have information of the topology of the network.

The SensorChannel class decides the nodes that should receive the stimuli depending on the propagation model and the channel properties. We have implemented Seismic Propagation and Acoustic Propagation.

The Seismic propagation model calculates the received signal power as a function of distance between sender and receiver and the attenuation factor. The received signal power P_r is calculated as

$$P_r = \frac{P_t}{\max(d, d_0)^{f_a}}$$

where

P_t : power with which signal transmitted

d : distance between sender and receiver

d_o , f_a : signal attenuation factor can be configured

In Acoustic Propagation, the received signal power Pr is calculated according to the following equation

$$P_r = N(p \times \mu_g, \sigma_g^{\ 2})$$

where

$$p = \frac{P_t}{\max(d, d_o)^{f_a}}, \mu_g = U(\min_g, \max_g)$$

P_t : power with which signal was transmitted

d : is the distance between sender and receiver

$\min_g, \max_g$, μ_g , σ_g : min, max, mean and variance of microphone gain

d_o , f_a : signal attenuation factor, can be configured

5.3 Sensor Node

The SensorNode module is a compound module that has the different layers of the protocol stack as the sub-modules. The sensor Node module definition and the class represent all the components of the sensor node.

5.4 Coordinator Module

Coordinator class has the functionalities that coordinate the activities of the hardware and the software modules of the sensor node. The Coordinator need to be extended and functionality added for access to properties of new hardware or consumers added. The Coordinator class has the reference to all the layers in the sensor node and all the layers in the sensor node may access the Coordinator. Thus through the Coordinator any layer may access and update the properties of the other layer. For example the battery needs to be informed on transmission or receiving packets and the energy consumption updated at the node. The Coordinator class is responsible for registering the sensor node to the sensor network. Registering of the sensor node is an indication that the sensor node is up and functioning. On complete energy depletion the node is unregistered from the sensor network. Typically a sensor network has the Radio, CPU and Battery registered with the Coordinator module. Various events in the network trigger the Coordinator to update the modules registered with it.

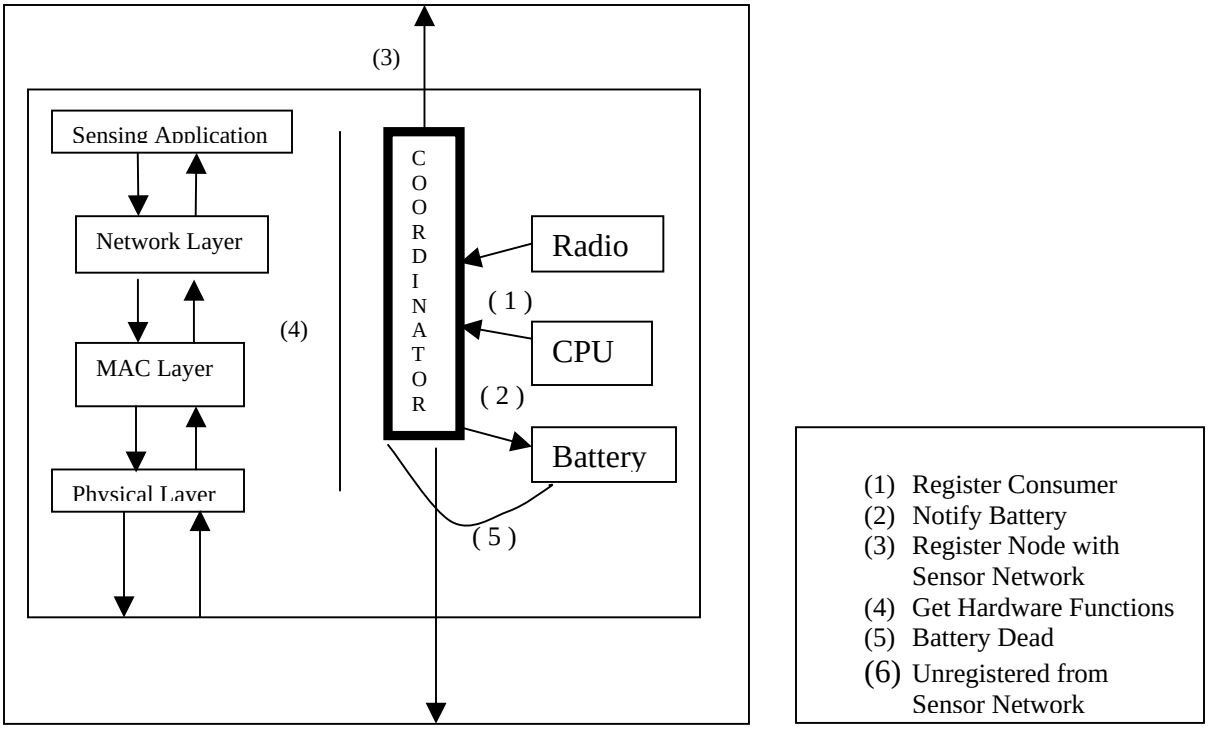


Figure 7 Nodes in a Sensor Network

5.5 Hardware Modeling

5.5.1 Battery Module

BatteryBase is the abstract class which is subclassed implementing the battery functionality. BatterySimple is a subclass of BatteryBase. It updates the energy consumption of the battery depending on the activities of the node. Energy consumption rate and operations may be extended in the battery model.

5.5.2 CPU Model

CPU Base is the abstract class for the different CPU models. CPU Simple has implementation of the power consumption of the CPU in different states: idle, sleep and active.

5.5.3 Radio Model

RadioBase is an abstract class for the different Radio models. Radio Simple a subclass of RadioBase updates the energy of the battery depending on the state of the Radio: idle, sleep, transmit, receive.

The values for the different properties of the hardware and consumers may be provided through the configuration file.

5.6 Software Model

The software model represents the different layers of the wireless protocol stack:

Sensing Application Layer: implements the application specific functions and other in-network processing depending on the application simulated such as aggregation and pass the result on to the network layer. The Sensing Application Layer receives the stimuli from the TargetNode through the sensor channel and takes appropriate action.

Network Layer: implements the routing protocol for sensor networks. Directed Diffusion and Geographically aware routing protocol have been implemented in this layer. The network Layer receives the message from the application layer, and then transforms the message to a macPacket type message and sends it to the bottom layer to the MAC layer. The NetworkPacket may be broadcast or unicast to specific node (sink node).

MAC Layer: The MAC_802_11 and Simple Mac implementation has been done at this layer. The macType message received form the above layer is sent to wirelesschannel through the PhyLayer that in turn interacts with the radio model

to transform the state of the radio before sending the message to the wirelesschannel. Energy is updated at regular intervals in the node as and when the different consumers change state.

5.6.1 Wireless Channel

The Wireless Channel Module controls and maintains all potential connections between the Sensor Nodes. These static connections are provided from all the nodes to the Wireless Channel Module and from the module to all the nodes in the NED file. These connections enable Sensor Nodes to exchange data and communicate with each other. Any message from a node is sent to all the neighbors within its transmission region with a delay d where d is (Distance between the communicating Sensor Nodes) / Speed of Light.

Various Radio Propagation models are used to predict the received signal power of each packet. These models affect the communicating region between any two nodes and are derived by the Wireless Channel.

Free Space Propagation Model: The free space propagation model assumes the ideal propagation condition that there is only one clear line-of-sight path between the transmitter and receiver. H. T. The received signal power in free space at distance from the transmitter is estimated as: [25]

$$P_r = (P_t * G_t * G_r * \lambda^2) / (4\pi)^2 * d^2 * L^2$$

- P_t is the transmitted signal power
- P_r is the received signal power
- G_t, G_r are the antenna gains of the transmitter and the receiver respectively.

- L is the system loss, and λ is the wavelength.

Two-ray ground reflection model: A single line-of-sight path between two mobile nodes is seldom the only means of propagation. The two-ray ground reflection model considers both the direct path and a ground reflection path. This model gives more accurate prediction at a long distance than the free space model. The received power at distance d is predicted by

$P_r = (P_t * G_t * G_r * h_t^2 * h_r^2) / (d^4 * L)$

h_t and h_r - heights of transmit and receive antennas respectively

The above equation shows a faster power loss than for Free Space Model as distance increases.

Chapter 6 : Implementation Details for LSU-SensorSimulator

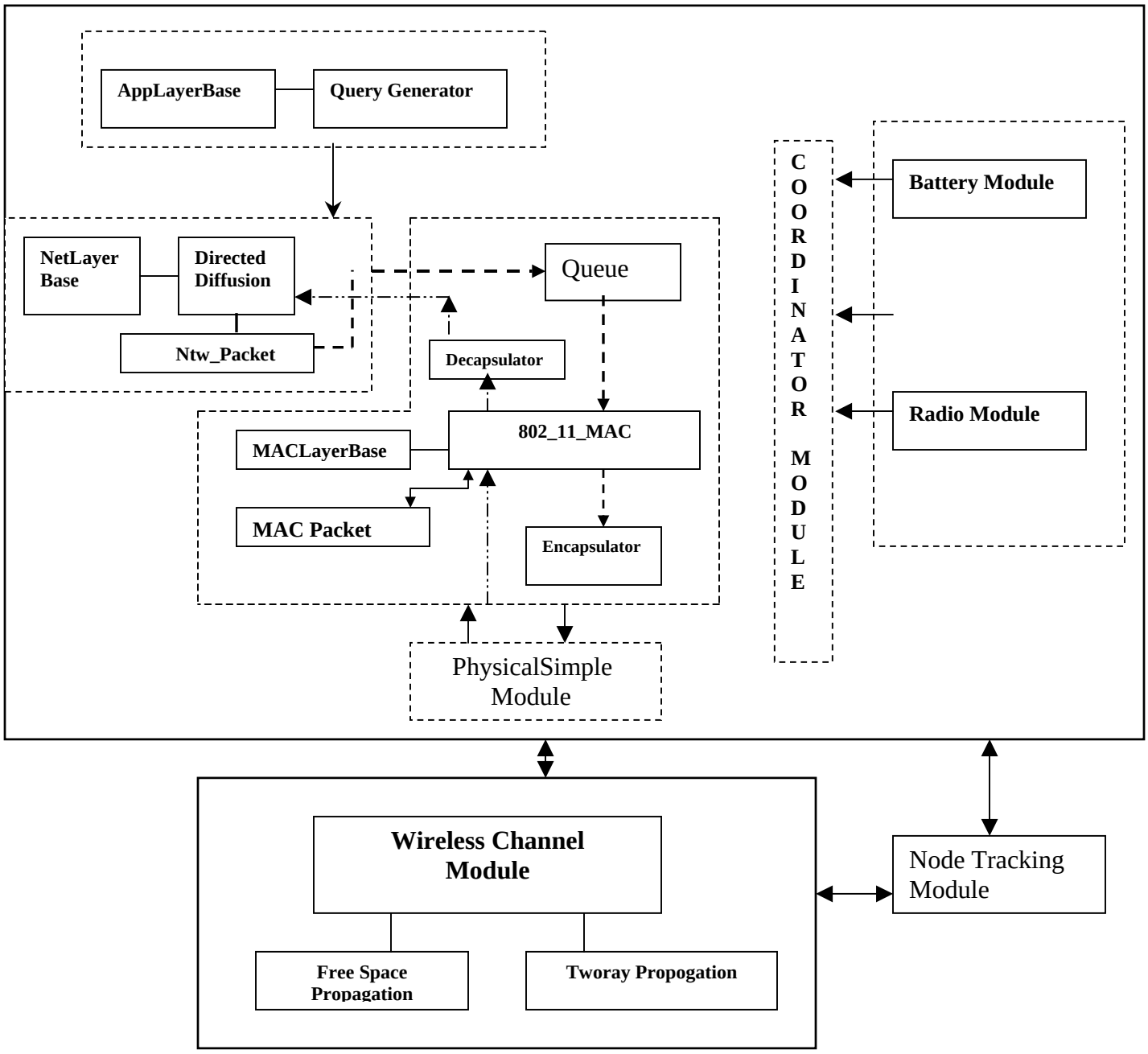


Figure 8 High Level Design for LSU-SensorSimulator

6.1 Directed Diffusion with GEAR Implementation

We have implemented Directed Diffusion along with Geographic Routing. The Application Layer generates interests that specify the region, the kind of data

required and rate of delivery of data. The Query message contains attributes, rate of data and duration. The attribute structure has features to specify interest properties such as the region of interest and any user-defined query messages.

- Query→attribute
- Query→rate of data
- Query→duration

These nodes that initiate the interest messages are called Subscribers. On receiving the interest message, the network layer broadcasts the beacon messages in the network. The immediate neighbors of the node on receiving beacon messages reply back with beacon-reply type of message that contains their geographic location and the energy left in them. The node waits a period of time to receive the beacon-reply from its neighbors. The interest message is then forwarded to the node that has a higher estimated cost to the region as calculated by the GEAR protocol. The next node follows the same procedure and forwards the message towards the region by Geographic Routing. If a node in the path does not have any neighbors or all its neighbors are away from the region, then it sends a message to its parent node that it is a dead-end. The parent node on updating the cost of the unreachable node, forwards the query in an alternate route towards the region. In the specified region, the interest is recursively flooded. The interest cache is maintained at each of the nodes in the path with its gradient of interest to each of the neighbors. The nodes in the region that have the specified properties of the interest send out data. These nodes are referred to as Publishers.

The data is marked as Exploratory to reinforce the path that was taken by the interest. On receiving the data marked as Exploratory by the subscriber, positive reinforcement message is sent out by the Subscriber node. Each node on path forwards this message thus reinforcing the path to the region. When the node reinforces a path, its cost to the region is known and this cost is sent back to its source node, which updates the cost information of that node to the particular region of interest. Thus the path with the highest cost is always maintained, reinforcing the route. The data from the region follow the path established by the reinforced messages. The nodes in the region send out data at the rate that is specified in the query. Data caching is implemented in intermediate nodes and so the data requested by different subscribers from the same region may be satisfied by the common node in the path thus reducing the traffic and redundant messages. The data marked as exploratory are sent to identify better paths and reinforce at regular intervals. Also the neighbor-updating procedure phase is carried out, i.e. at regular intervals the beacon messages are broadcast and beacon-reply messages are sent by neighbors thus maintaining latest neighbor information.

6.2 MAC 802.11

The MAC layer places the network packet on the Wireless Channel. The NetworkPacket may be a broadcast or unicast packet to a specific node (sink node). Any network layer packet received by the MAC-802-11 [24][25][26] module is encapsulated into MAC frame with the MAC header added to it. The Network layer packets have the information whether the packet has to be

broadcast or unicast. Broadcast packet is encapsulated into Broadcast MAC frame with appropriate MAC Header and is put in the Messages-queue of the MAC Layer. If the Network packet is for a particular destination, RTS frame is created and is inserted in the Messages-queue of MAC layer. If the Network packet length is more than the MAC frame, it is fragmented and the fragments for that Network Packet are created with MAC headers and are inserted into the Fragments Queue. The MAC layer then waits for the channel to be idle to send its frame from the Messages-queue. MAC layer has a NAV Timer, which specifies the busy/idle state of the medium. NAV Timer set for a node implies that the channel is busy. When the NAV Timer expires the MAC layer waits for the channel to be free for DIFS time and if the channel is still idle after DIFS timer gets expired, it then goes into Exponential BackOff. It then waits for a random time set by the BackOff Timer. The BackOff Timer decrements its value during the idle period of channel. The node whose BackOff Timer expires earlier will get the chance to transmit its next frame. All the intermediate nodes receive this frame, set their NAVTimer to the value obtained from the Header field of the received frame. Then the BackOff Timer of the intermediate nodes is stopped from decrementing. Once the channel becomes idle (when the NAVTimer expires) all the nodes start decrementing their BackOff Timer. The node whose Back Off Timer expired earlier and got the channel will send the first message from the Messages Queue. If it is a broadcast message, then all the nodes in its region receive it and the MAC layer of those nodes decapsulate the Network packet and send it to the Network Layer. If it is a RTS frame, the Destination

node checks whether its NAV timer is set or not (its transmission region is busy or not) and then responds to it by sending CTS. All the other intermediate nodes receiving this RTS update their NAV Timer to the CTS+DATA+ACK duration which implies that the channel is busy for that duration and hence refrain from transmitting during this interval. If the Destination node receives more than two RTS requests within a time interval then collision occurs and the Destination node does not respond (send CTS) to any of these RTS requests. The Source node which is sending RTS have an RTSExpired Timer set for RTS frames, when they are sent to the Destination node. This timer is scheduled to expire after RTS+CTS duration. If the Source node does not receive CTS within this duration, RTSExpired Timer gets expired and retry counter of that RTS frame is incremented. If the retry counter is less than ShortRetryLimit (as per the specification), then the Contention Window is doubled and the random time set by the BackOff Timer is chosen between 1 and the Contention Window size. If the retry counter reaches ShortRetryLimit, then the message (RTS and corresponding Fragment) is dropped by the MAC.

If the Destination node responds to RTS by sending back the CTS, the intermediate nodes for CTS will update their NAVTimer obtained from the Header field of CTS frame (Data+Ack duration) and hence refrain from transmitting during this interval. Once the Source node gets the CTS, it will send the corresponding fragment of the Network Packet to the Destination and waits for an Acknowledgement. The Destination node upon receiving the Data frame extracts the Network packet, sends it to the Network layer and sends back the

Acknowledgement to the Source node. Once the Source node gets the Acknowledgement it checks and sends if there are any other fragments to be sent to this node without any additional RTS frames.

Chapter 7 : Experimental Setup for LSU-SensorSimulator

In this section we present the results from the comparative simulations run on ns-2 and LSU-SensorSimulator. Exactly same simulations are run on both the simulators and the results are then compared. The simulators are compared for simulation time, memory utilization and scalability.

The same random coordinates distributed over the same grid are used for the nodes for both the simulators. The queries generated for both the simulations are the same. The region of interest has the same number of nodes for both the simulators, as these effects the flooding and data messages. The other factors which can affect the simulation results are kept same, like the neighbor update time, battery power, path re-enforcement time etc. Both the simulations run the simulations for the same duration and for the same CPU time.

To create a very generic test scenario N sensor nodes are randomly placed in a region of MxP size. Randomly few nodes send queries towards a region of interest. The path taken by queries is decided by first sending interests. We have implemented attribute list to define type of interest or data message. When a node receives an interest message, it first checks if it has the property list of its neighbors. The property list that the node maintains is the distance from the neighboring node to the final destination and the energy levels of the neighboring nodes. If the node has this list, it checks the last updated time of the neighbor list. If this time is within the permissible limit, this information is used to decide the next hop neighbor. If the neighbor list does not exist or the last updated time is more than the desired time limit then beacon messages are sent

out. All the neighboring nodes receive this beacon message. The neighboring nodes then send back beacon reply messages, which update these properties in the neighbor list. The next hop neighbor decision is based on the GEAR protocol specifications. The next hop is decided based on the higher value of $c(N_i, R) = \alpha d(N_i, R) + (1-\alpha)e(N_i)$ as described above. For our implementation we have given equal weightage to distance and energy factors. After the query reaches the region of interest, it is flooded to all the nodes in the region. A visited node list is maintained to avoid going into a loop. When a node in the region of interest receives an interest it sends back an exploratory message to the source of the interest. The exploratory message follows the reverse path taken by the interest message. It gets the reverse path information from the nodes. When this exploratory message reaches the source node, the source node reinforces the path by sending back reinforcements. The reinforcements might or might not follow the same path as the initial interest message. On the arrival of the reinforcements the nodes in the region of interest start sending back data messages at the rate specified in the interest. At regular intervals these data messages are marked as exploratory. When the source receives a data message marked as exploratory it sends reinforcements to rebuild the path. This would take care of any holes that might have been formed in the path.

In order to test the performance of the simulation framework we ran the setup with queries generated by 10 nodes at random locations in the network. A similar test was performed with 100 nodes generating queries. The queries follow a multi-hop route to the region following the procedure mentioned above. Once

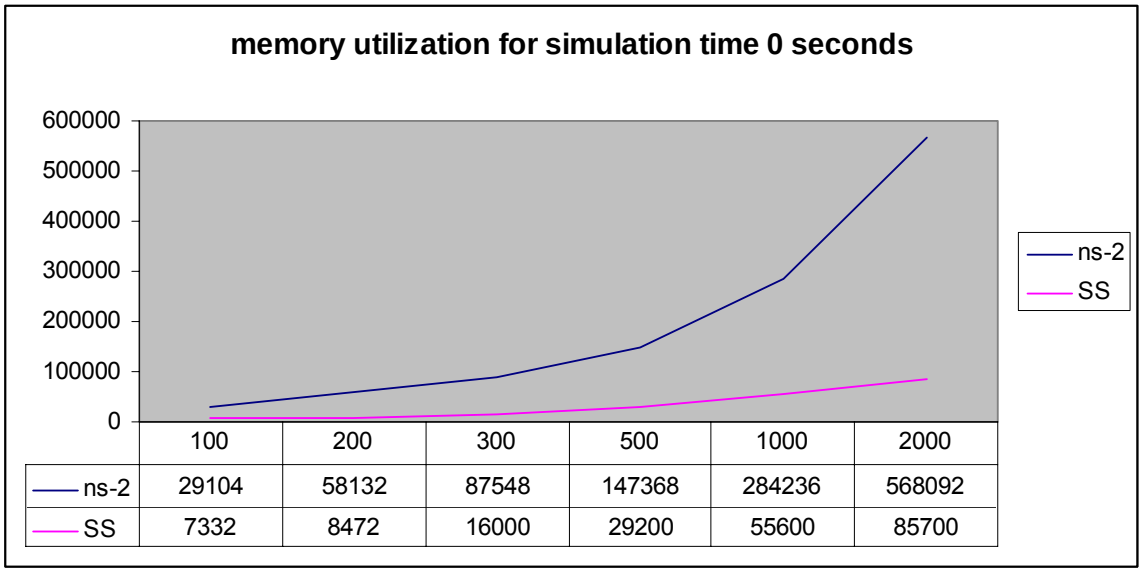
the query reaches the region the data is sent back once every 5 seconds for the complete simulation time by all the nodes in the region. The objective of this kind of setup is to check whether the simulation framework is able to handle the traffic generated and run to completion as well as to check the amount of time required to run the simulation. Figure 10 and 11 show the performance of the two simulators (ns2 v/s Sensorsimulator) for the setup with 10 nodes and 100 nodes generating queries. It was observed that the performance of both the simulators ns2 and SensorSimulator showed similar results for less number of nodes in the network. As the number of nodes in the network increases, SensorSimulator is able to handle the traffic and the events generated in a better fashion so as to complete the simulation in a reasonable time faster than ns2.

The simulator performance is affected by traffic generated by the messages flowing. The traffic generated varies due to many factors. The number of sensor nodes is one of the prime factors to this effect, but not always the reason for the traffic. The following factors influence the traffic collectively.

- Number of nodes in the grid
- Number of nodes in the region of interest
- Number of queries
- Duration for which the query will be active
- Grid size
- Node density in the region and the range of the nodes

Number of nodes in the grid causes the maximum impact at the start up time, as the memory usage increases as the number of nodes in the system

increases. To compare the two simulators for the performance for only the scalability for the number of nodes we ran simulations on the both the simulators for different number of nodes for 0 sec simulation time. This in effect means that the whole simulation environment is loaded by the simulators, and this is the only factor utilizing the system resources, as no other activity happens in the simulations. The results are as follows



	100	200	300	500	1000	2000
ns-2	29104	58132	87548	147368	284236	568092
SS	7332	8472	16000	29200	55600	85700

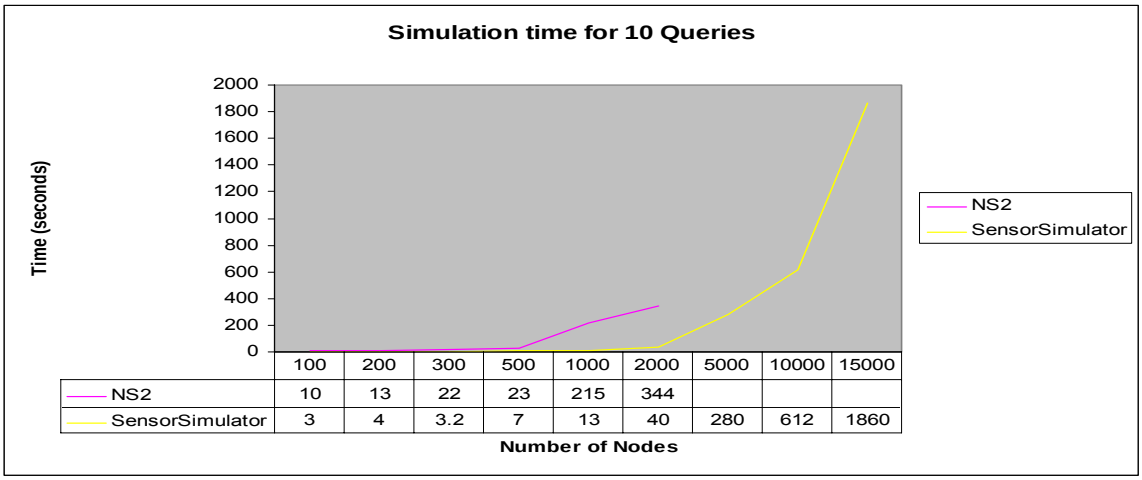
Figure 9 memory utilization graph

These comparative results show that LSU-SensorSimulator has a much better memory management for large number of nodes. ns-2 memory usage is very high for large number of nodes, which has two major effects on the simulation :

- More start up time for ns-2 as compared to LSU-SensorSimulator
- Higher memory requirement to load a larger number of nodes.

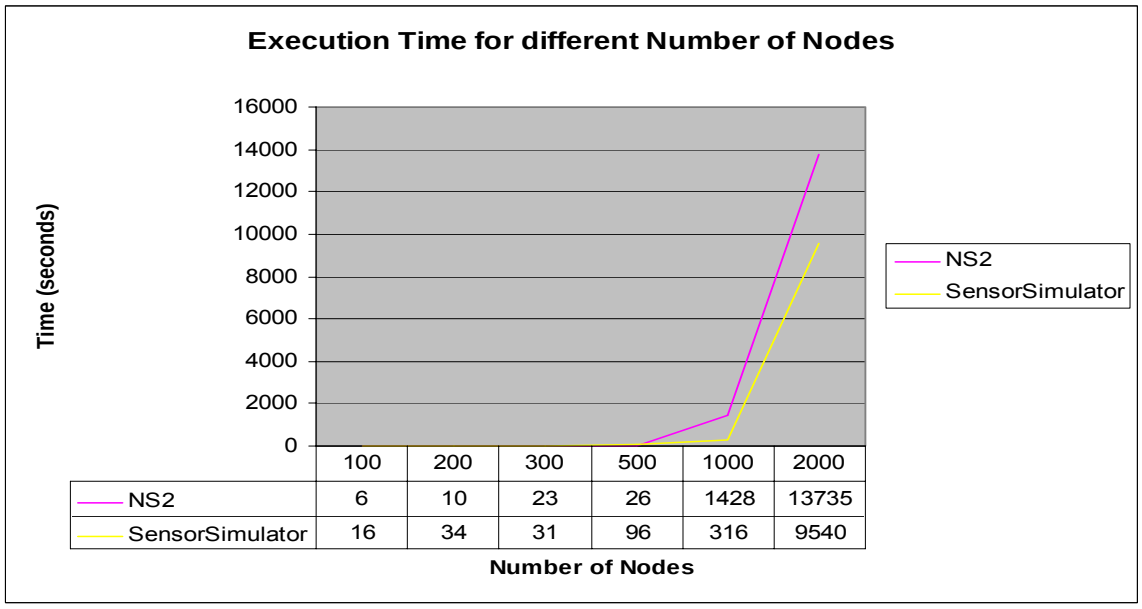
To compare the performance of the simulators we compare the time taken to run two simulations under the same traffic conditions. We compare the results

for two cases. Figure 10 compares the results for different number of nodes and 10 queries, and figure 11 compares the results for different number of nodes and 100 queries. All the other factors which influence the traffic such as number of nodes in the region of interest, node density and the time duration for which a query is alive are same for both the comparisons.



	100	200	300	500	1000	2000	5000	10000	15000
NS2	10	13	22	23	215	344			
SensorSimulator	3	4	3.2	7	13	40	280	612	1860

Figure 10: Performance ns2 v/s SensorSimulator for 10 queries



	100	200	300	500	1000	2000
NS2	6	10	23	26	1428	13735
SensorSimulator	16	34	31	96	316	9540

Figure 11: Performance ns-2 v/s SensorSimulator for 100 queries

Chapter 8 : Results for LSU-SensorSimulator

The simulations are run for the number of nodes specified in the table. The range of transmission region is maintained as 35. Region sends data for every 15 sec till the simulation ends. The simulations are run for the same topology, timing parameters with Directed Diffusion and MAC-802.11 implemented at Network Layer and MAC Layer respectively in both ns2 and LSU-SensorSimulator. The path taken in both of them is verified to be similar.

The Simulations are run for 145 sec for smaller networks and for 300 sec for 100 and 200 node networks.

Table 1 Implementation Comparison for GEAR and Directed Diffusion

No:of Nodes	**Network Size**	**No:of Queries**	**No:of nodes in region**	**Region Boundary**	**Data Generated (By region) OMNeT++**	**Data Received by query node**	**Data Generated – ns2**	**Data Received - ns2**	**Time (secs)**
5	150-150	1	2	5 – 5	35	35	48	48	145
10	150 – 150	1	2	5 - 5	36	36	48 - 48	48	145
50	200 - 200	3	2	10 - 10	34	34	3: 30 4: 54 5: 30	3: 30 4: 54 5: 30	145
100	200 - 200	10	5	20 – 40	6 – 45 7 – 40 8 – 40 9 – 40 10 – 40 11 – 70 12 – 40 13 – 40 14 – 70	6 – 35 7 – 39 8 – 39 9 – 39 10 - 39 11 - 59 12 – 39 13 – 39 14 – 60	6 – 50 7 – 70 8 – 70 9 – 70 10 – 50 11 – 90 12 – 70 13 – 70 14 – 90	6 – 49 7 – 63 8 – 70 9 – 63 10 – 50 11 – 82 12 – 70 13 – 63 14 – 82	300

Table continued

200	200 – 200	10	5	20 – 40	6 – 30 7 – 30 8 – 30 9 – 30 10 – 45 11 – 65 12 – 30 13 – 30 14 – 65 15 - 30	6 – 29 7 – 29 8 – 29 9 – 29 10 – 37 11 – 57 12 – 29 13 – 29 14 – 57 15 - 29	6 – 70 7 – 70 8 – 70 9 – 70 10 – 50 11 – 90 12 – 70 13 – 70 14 – 90 15 - 70	6 – 66 7 – 66 8 – 66 9 – 66 10 – 45 11 – 90 12 – 66 13 – 66 14 – 90 15 - 66	300

The total number of data packets generated by region for each node is specified in the table. And the number of data packets received by them is also listed. For smaller networks, both the simulations achieved 100% delivery ratio. For 100 nodes, both ns2 and SensorSimulator achieves 90% delivery ratio. For 200 nodes, ns2 shows delivery ratio of 93.6% and SensorSimulator shows 92.8% delivery ratio. The simulations prove that SensorSimulator shows similar behavior with ns2 in its implementation, to achieve better performance and memory requirements.

The main requirements from LSU-SensorSimulator were

- Scalability, as the sensor networks typically consist of thousands of nodes, the simulator is able to simulate a network with large number of nodes. It was seen that on similar system resources ns-2 could simulate till 2000 nodes, after that the CPU usage would go above 99% and the systems hangs. But with LSU-SensorSimulator, I was able to simulate for 15000 nodes.

Table 2 Nodes in the Network

Ns-2	2000
LSU-SensorSimulator	15000

- Simulation time is another factor. Since the simulations use lot of system resources, to simulate a 1200 second simulation the simulator might take a long time. LSU-SensorSimulator was designed to simulate much faster by better memory and resource management.

Table 3 Simulation Time Comparison between ns-2 and LSU-SensorSimulator

	1000 nodes 10 queries	**1000 nodes 100 queries**	**2000 nodes 10 queries**	**2000 nodes 100 queries**	**5000 nodes 10 queries**	**3000 nodes 100 queries**	**15000 node 10 queries**
ns-2	215	1428	344	13735	Does not complete	Does Not complete	Does Not complete
LSU-SensorSimulator	13	316	40	9540	280	17800	1860

A uniform node density was maintained for all the experiments. Also the number of nodes in the region of interest was constant for all these simulations.

- Memory utilization for simulations in which no queries are sent out and the memory utilization for simulations with random queries sent out to random regions show how well the simulators can manage the system resources. The better the memory utilization, the better would be the performance.

Table 4 Memory Utilization Comparison between ns-2 and LSU-SensorSimulator

	ns-2	**LSU-SensorSimulator**
0 queries 500 nodes	147368	29200
0 queries 1000 nodes	284236	55600
0 queries 2000 nodes	568092	85700
10 queries 2000 nodes	569480	96400

The second part of this thesis work describes “Buddy load sharing routing protocol” which aims to increase the network life.

Chapter 9 : Buddy Load Sharing Routing Protocol

One of the important requirements of any network is connectivity and longer survivability of un-partitioned network. Due to the limited energy the network topology in WSN is very dynamic. If nodes in the network have a more uniform energy consumption then the network would die out gracefully. Also the nodes in the network continue to provide connectivity for longer time, and the time to network partition increases.

In a typical WSN sensor nodes collaborate with each other to pass query and data messages from one part of the network to the other. A typical sensor node consists of a base station which sends queries to the region of interest. After receiving the query the nodes from the region of interest start sending back data message to the base station. Since the nodes are typically distributed over a large area and the range of each individual node is not much, the messages hop from various nodes before reaching their desired destination. The route a message takes depends on the routing protocol implemented in the networking layer. Since energy is a very constrained resource in sensor networks, the routing protocols try to take a path which is most optimal for energy consumption. But as the lowest energy path might not be optimal for network connectivity. We propose a routing protocol which takes care of the network connectivity. This protocol can be combined with another protocol to suit both energy and connectivity problems.[27]

Due to finite energy source of the sensor nodes the network topology keeps changing very frequently. The nodes which participate in more number of

routes for transmitting data will die out faster than other nodes. Thus the network after some time becomes very unbalanced. That is parts of the network are not accessible as the nodes on the route to that part are very low on energy. This is network partitioning, and this leads to shorter network life.

To increase the network life and network connectivity we propose a routing algorithm, which identifies nodes which are most likely to have maximum amount of traffic. By using this algorithm for routing the data and query messages the load is distributed amongst the nodes, thus having a more balanced energy usage. This contributes to increasing the network life. Figure 12 and 13 show a typical sensor network that uses a geographic routing protocol. The size of the nodes indicates the amount of energy left in the nodes. A bigger node size indicates that the node will probably last for a longer time as it has more power left. The yellow node indicates that the node is very low on power and thus not capable of transmitting any data. The WSN network in Figure 12 sends out random queries to randomly selected regions from the base station (indicated by the red node). The routing protocol implemented in the network layer is GEAR with Directed Diffusion. Figure 12 illustrates the network topology after N seconds of simulation time. It can be observed that there are many high energy nodes (indicated by big circles) and shows many energy depleted nodes (indicated by small yellow circles). The region indicated by the dotted circle is no longer reachable as all the nodes near the region are too low in energy to transmit any more messages. The figure 13 shows another WSN with randomly distributed nodes. Random queries to randomly chosen regions are sent out. The

routing protocol implemented in the network layer is GEAR with Directed Diffusion along with buddy load sharing routing protocol. Figure 13 illustrates the network topology after N simulation seconds. There are very few High energy or low energy nodes around the base station. The nodes around the base station have similar energy levels. Thus the network connectivity is much better in this case.

In a sensor network the topology of the whole network is not known. Since the processing power of the sensor nodes is limited, it is not possible to identify the nodes which would lead to the network partitioning. So we propose an alternating algorithm. There are few nodes located in the range of the controller. Any data flowing to and from the controller will have to go through these nodes. If the routing protocol can make sure equal distribution of load to all these nodes, this would greatly increase network survivability. This protocol tries to elongate the network connectivity time by trying to maintain a balance of energy health in the nodes which are used more frequently than any other node in the network. This protocol can be used with any other routing protocol and any number of nodes can be included in the alternating protocol. The protocol can be extended to include one hop neighbors, two hop neighbors or n hop neighbors of the controller, depending on the need for network connectivity. This algorithm tries to bridge the wide disparity in the energy levels of the nodes which lead to the network partitioning. If the load is evenly divided between the nodes most frequently used the network would be able to provide better connectivity. Thus

leading to a graceful degradation of the network, and a very limited overhead on the nodes.

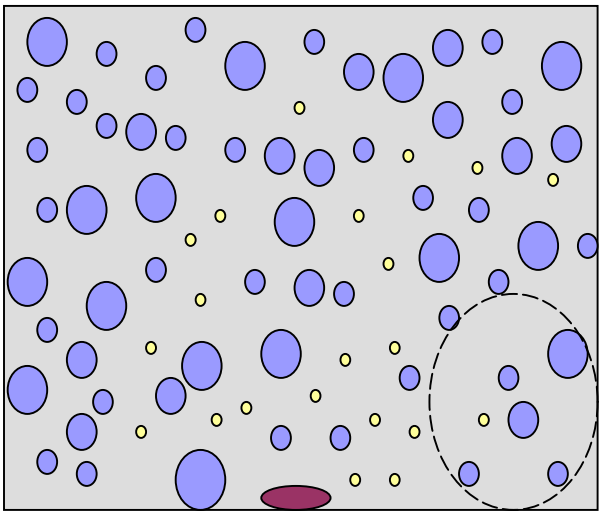


Figure 12 WSN without Load Sharing

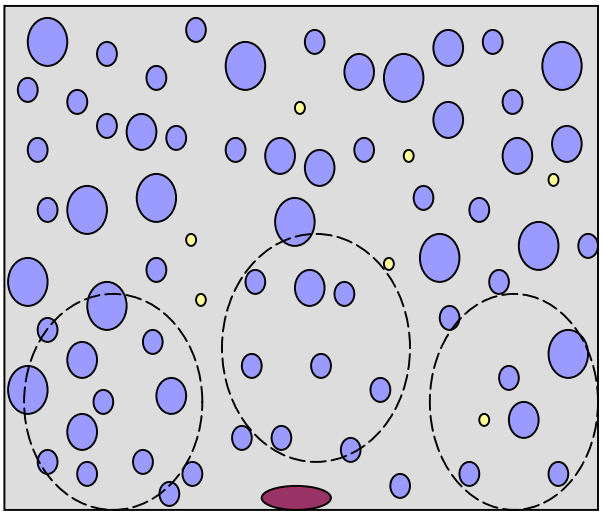


Figure 13 WSN with Load Sharing

The algorithm is based on the fact that some of the nodes around the base station would handle much more traffic than other nodes in the network. Thus if the load can be distributed evenly for these nodes we can have better connectivity. The first step is to identify the regions in which nodes are likely to handle a significantly higher number of messages as compared to other nodes in the network. The nodes in the network which are placed closer to the controller will be used more than most of the other nodes in the network. Thus by marking these nodes and distributing the load evenly between them the network connectivity can be increased. To illustrate the algorithm let us take a typical sensor network Figure-14 shows a typical sensor network. Now if the controller sends out a query for region A. The underlying routing protocol implemented in the network layer can quite possibly route the query from the controller to region A through node 1 to node 7 and then to region A. If the query requires that data

be sent for long enough duration the battery of node 1 will eventually drain out. If the controller sends out another query for region B. Region B could be possibly located such that it can only be connected to controller through node 1, as node 6 is out of range for most of the other nodes. This would lead to network partitioning as there is no way region B can be connected once the battery of node 1 drains out. Since the battery for all the nodes in the network is limited this would eventually happen in the network, but if the importance of node 1 is known to the routing protocol it can try to use other nodes instead of node 1 whenever possible and thus increase the network life.

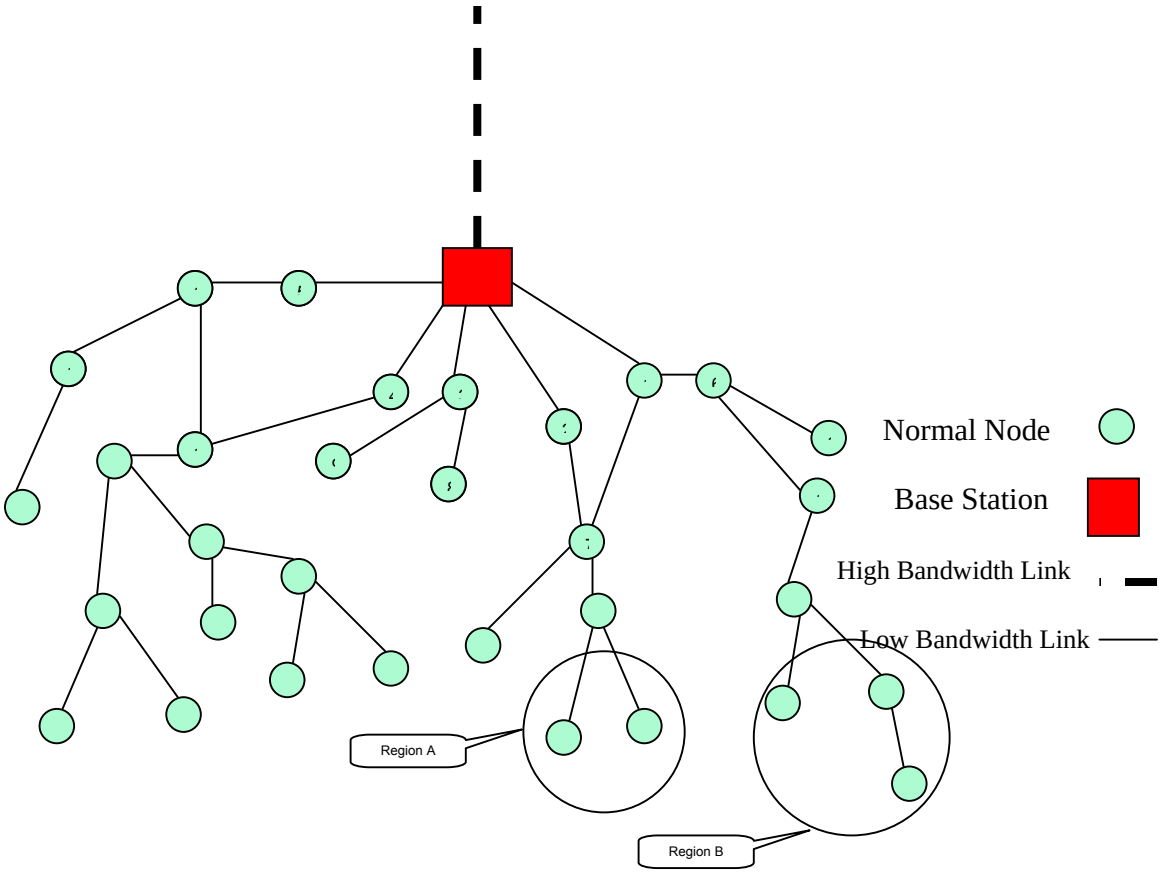


Figure 14 Wireless Sensor Network

Chapter 10 : Algorithm for Buddy Load Sharing Routing Protocol

10.1 One Hop Buddy Load Sharing Algorithm

Before the base station sends out queries to a region, it first sends out beacon messages to its one hop neighbors, telling them that they are likely to have more traffic since they are in its neighborhood. These nodes are referred to as cluster nodes. This identification helps the buddy load sharing algorithm to distribute work evenly between all these important nodes. After this marking of nodes is over the base station sends out the query to the desired region (or to the whole network, as the case might be). The path of the query is decided based on the underlying routing protocol implemented by the network layer. The Buddy load sharing algorithm does not influence this path selection in any way. When the query goes from a cluster node to a regular node, to the receiving node the query packet indicates that it has come from a cluster node. These nodes are marked as secondary cluster nodes. That is to indicate to the current node that when sending back data packets distribute the load evenly. These secondary cluster nodes keep a list of all the one hop cluster nodes in its neighborhood. It makes this list by sending out beacon message asking all the nodes in its neighborhood which are cluster nodes to reply. When data packets reach the secondary cluster node, the messages no longer follow the route indicated by the routing protocol, but are distributed amongst the one hop cluster node. Thus facilitating a graceful depletion of energy levels in the important

nodes in the network, leading to longer network connectivity. This algorithm can be applied along with any geographically aware routing protocol.

10.2 N Hop Buddy Load Sharing Algorithm

The base station marks all the nodes in its N hop neighborhood, by sending out beacon messages which are recursively sent out till N hops from the base station. The N+1 hop node from the base station on receiving the query message is marked as the secondary cluster node. While sending back data packets all the cluster nodes in the neighborhood of secondary cluster node equally share the load. And they in turn distribute it evenly to their neighboring cluster nodes.

The following WSN illustrated the one hop buddy load sharing algorithm. The routing protocol used is GEAR with directed diffusion. The 7 nodes are distributed as shown in the figure 15.

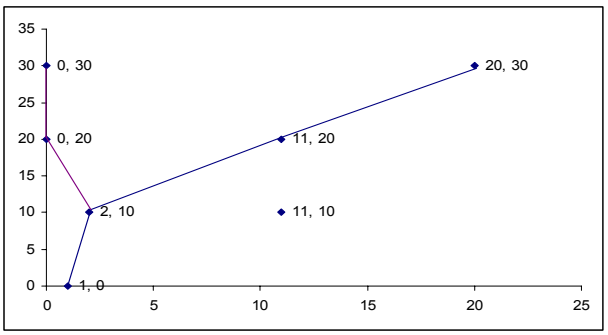


Figure 15 WSN without buddy load sharing

First take the case when the distribution of load in important nodes does not take place. The first query Query1 goes from node at (1,0) to the region around node (20, 30) and after some time the second query Query2 goes from

node (1,0) to the region around node (0, 30). The path taken by query and the data packets for Query1 based on GEAR with directed diffusion is

(1,0)—(2,10)—(11,20)—(20,30)

As shown by the simulation results, indicated in figure 2 by blue lines. If the initial energy level of the battery is 2000 and each data packet going through a node consumes 20 units of battery power, then after 70 data packets have gone back through the node the battery would be considerably drained. At this instance there is another query Query2 from (1,0) to (0,30). This query also follows the GEAR and directed diffusion protocol.

The path it takes is

(1,0)—(2,10)—(0,20)—(0,30)

But since the node (2,10) is already depleted in battery power the node soon dies. After the node (2,10) dies there is no path which the Query2 can take to get data from the region around (0,30). Thus we have network partitioning.

Now we consider the same network and the same routing protocol (GEAR and directed diffusion) along with the load distributing algorithm. The first query Query1 goes from node at (1,0) to the region around node (20, 30) and after some time the second query Query2 goes from node (1,0) to the region around node (0, 30). The path taken by query and the data packets for Query1 based on GEAR with directed diffusion is

(1,0)—(2,10)—(11,20)—(20,30) and (1,0)—(11,10)—(11,20)—(20,30)

Before the query is initiated the controller node which in this case is (1,0) marks the nodes within the first cluster. The same in this case are (2,10) and (11,10).

When the Query1 reaches the (11,20) node it is also marked as the secondary cluster node. Data packets start going back from region around (20,30). When they reach (11, 20), beacon messages are sent out to get all the marked nodes which are neighbors of (11,20). In this network both (2,10) and (11,10) are neighbors of (11,20). So the data packets are alternatively sent to (2,10) and (11,10). As shown in figure 16. Thus ensuring that even after 70 data packets the node (2,10) is nearly only half as depleted as in the case without the load distribution.

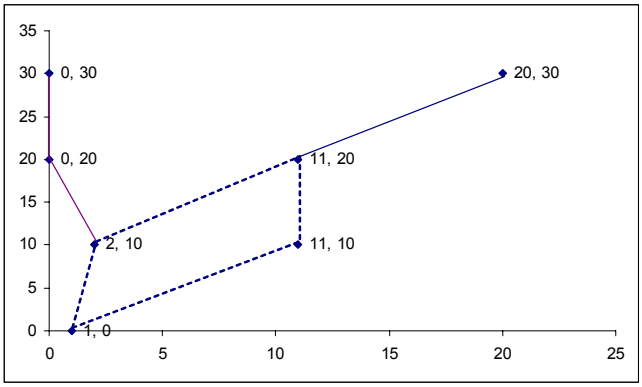


Figure 16 WSN with buddy load sharing

In the next chapter the network connectivity and life are compared with and without using buddy load sharing routing protocol.

Chapter 11 : Experimental Setup for Buddy Load Sharing Routing Protocol

To compare the network life and connectivity of a random WSN with and without using buddy load sharing routing protocol a few randomly generates networks with 1000, 2000 and 5000 nodes and 10 and 100 queries are used. The buddy load sharing routing protocol can be integrated with any routing protocol, for my simulations, I have used GEAR and directed diffusion. The results are compared by using GEAR and Directed Diffusion along with Buddy load sharing algorithm and without it.

11.1 Directed Diffusion

Directed Diffusion is a data-centric information dissemination paradigm for Wireless sensor networks. The elements of directed diffusion are sending interests, setting up gradients, and reinforcing the paths. An interest message is a query that has the information about the data that is required from the sensor nodes. Data can be either collection of information or an event triggered by some physical phenomena. Gradients are directional state created in each node, set towards the neighbor from which interest is received. One or more of these paths are reinforced. Each task is named in an attribute list. The task description specifies an interest for data matching. Interest is a named task. Interest is sent into the network from a sink. Interest may also have information about duration of the task and the interval at which response is required. Initial interest messages are also called Exploratory, and it tries to form a connection with the nodes that

have the required data. At each node a cache of distinct interests is maintained (this allows interest aggregation). They contain information about the previous hop. The interests propagate through the network. The nodes in the region or nodes that have data for a particular interest send data marked as exploratory through the gradient established. As a result exploratory data may follow multiple gradient paths to the query source node. Once the exploratory data is received, the query source node reinforces one of the paths based on the routing protocol being used. To reinforce the node sends a positive reinforcement message to the neighbor initiating the sending of data. The data sending interval is less than the exploratory sending interval. The reinforced neighbor reinforces its neighbor in turn, and this is done all the way till the data source. Data messages are marked as exploratory at a regular interval. Many other protocols have been proposed which are either improvement on directed diffusion or following similar concept.

11.2 GEAR

Geographical Energy Aware routing[16] uses a geographical and energy aware neighbor selection heuristic to route the packet towards the target region. The process of forwarding a packet towards the region involves

- choosing a neighbor that is closest to the destination among all the neighbors

when all neighbors are away, chose a neighbor that minimizes the cost value to the neighbor which is computed as

$$c(N_i, R) = \alpha d(N_i, R) + (1-\alpha)e(N_i)$$

where $d(N_i, R)$ is the distance from Ni to the centroid D of the region R normalized by the largest distance among all the neighbors Ni and $e(N_i)$ is the consumed energy at node Ni normalized by the largest consumed energy among the neighbors of N. On reaching the region of interest recursive forwarding technique is followed to flood the packet in the region to minimize the cost consumption.

Chapter 12 : Results

Since the energy is a finite source and the main cause of energy consumption is sending and receiving messages, the total number of messages sent are compared to estimate the energy overhead of using the Buddy load sharing algorithm.

Table 5 Comparison between the Total Number of Messages Sent

	With Buddy load sharing	**Without Buddy load sharing**
7	5469	5300
1000	7159212	7157604
2000	7906787	7911734

The results show a very little increase in the total number of messages sent in the two cases thus indicating a very small overhead in terms of messages(beacon messages), resulting in a small overhead for the energy consumed.

To compare the simulations for network connectivity, random queries to randomly selected regions is sent out continuously over a period of 150 seconds of simulation time. The following table compares the percentage of queries reaching the region of interest.

Table 6 Comparing the Connectivity of the Networks

	With buddy load sharing	**Without buddy load sharing**
100 queries	70%	60%
200 queries	62%	40%

The results indicate that even though the number of queries increase in the network, adding to the traffic and thus resulting in higher energy consumption

by the nodes, with buddy load sharing algorithm we can still achieve a much higher connectivity.

Chapter 13 : Conclusion and Future work

The first part of the thesis work provides a framework for wireless sensor networks. The second part of the thesis provides a routing protocol to increase the network life and connectivity. The framework has functionality for target node, sink nodes, sensor and wireless channel, power model.

Directed Diffusion and GEAR were implemented on this framework to demonstrate the use of the framework. The simulation results are compared with ns-2 simulation results for the same protocols under the same network properties. The results show that LSU-SensorSimulator can perform much better in terms of memory usage and simulation time. The buddy load sharing routing protocol gives approximately 20% increase in network connectivity.

References

1. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In Proc. of the ACM International Conference on Mobile Computing and Networking (ACM MobiCom'00), 2000.

2. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. IEEE/ACM Transactions on Networking, 11(1):2–16, February 2003.

3. J. Heidemann, F. Silva, and D. Estrin. Matching data dissemination algorithms to application requirements. In Proc. of the ACM Conference on Embedded Networked Sensor Systems (ACM SenSys'03), 2003.

4. Ns-2. http://www.isi.edu/nsnam/ns

5. SensorSim: A simulation framework for sensor networks. http://nesl.ee.ucla.edu/projects/sensorsim/.

6. S. Park, A. Savvides, and M. Srivastava. SensorSim: A simulation framework for sensor networks. In Proc. of the ACM international Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2000.

7. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. Computer Networks, 38(4):393–422, March 2002.

8. W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In Proc. 5th ACM MobiCom, August 1999.

9. Sobeih and J. C. Hou. A simulation framework for sensor networks in J-Sim. Technical Report UIUCDCS-R-2003-2386, Department of Computer Science, University of Illinois at Urbana-Champaign, November 2003.

10. H.-Y. Tyan and J. C. Hou. JavaSim: A component-based compositional network simulation environment. In Proc. of Western 23 Simulation Multiconference, CNDS'01, January 2001.

11. H.-Y. Tyan. Design, Realization and Evaluation of a Component-based Compositional Software Architecture for Network Simulation. PhD thesis, Department of Electrical Engineering, The Ohio State University, 2002.

12. OMNeT++, http://www.omnetpp.org/

13. http://www.intel.com/research/exploratory/wireless_sensors.htm

14. http://www.eng.auburn.edu/users/lim/sensit.html

15. J. M. Kahn, R. H. Katz, K. S. J. Pister: Next Century Challenges: Mobile Networking for “Smart Dust”

16. www.tinyos.net/

17. Gilbert Chen, Joel Branch, Michael J. Pflug, Lijuan Zhu, and Boleslaw K. Szymanski. SENSE : A SENSOR NETWORK SIMULATOR, Department of Computer Science, Rensselaer Polytechnic Institute

18. http://javasim.ncl.ac.uk/

19. http://www.ssfnet.org/

20. http://pcl.cs.ucla.edu/projects/glomosim/

21. http://www.scalable-networks.com/

22. Yan Yu, Ramesh Govindan, Deborah Estrin. Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks.

23. http://www.opnet.com/products/modeler/home.html

24. V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A Media Access Protocol for Wireless LAN’s. In ACM SIGCOMM 1994, 1994.

25. A. Tannenbaum. Computer Networks. Prentice Hall, 2002.

26. “Information technology - telecommunication and information exchange between systems – local and metropolitan area networks – specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications,” IEEE Standard, Tech. Rep., 1999

27. Rahul C. Shah and Jan M. Rabaey. Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. Berkeley Wireless Research Center, University of California, Berkeley.

28. C. Mallanda, A. Suri, V. Kunchakarra, S.S. Iyengar, R. Kannan, A. Durresi, and S. Sastry" Simulating Wireless Sensor Networks with OMNeT++ " , submitted to IEEE Computer, 2005

Vita

Ankur Suri was born in Kanpur, Utter Pradesh, India. She is the daughter of Mrs. Tripta Suri and Mr. Ram Paul Suri. She completed her undergraduate studies from Delhi University in 1997. She did her master's in mathematics from Indian Institute of Technology, New Delhi in 1999. She joined Louisiana State University, Baton Rouge, to pursue her master's degree in systems science in January 2004. Her research interest is sensor networks. She is a candidate for the degree of Master of Science in Systems Science to be awarded at the commencement of Dec 2005.