WIRELESS SENSOR NETWORKS

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

In

Electronics and Communication Engineering

Under the Supervision of

(Prof.Dr. Rajeev Kumar)

By

Harshit Sharma (111081)

Sahil Sharma (111092)

Saurabh prabhakar (111136)

To

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled "**wireless sensor networks**", submitted by *Harshit Sharma(111081), Sahil Sharma (111092),saurabh prabhakar (111136)* in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan, India has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Prof.Dr.Rajiv kumar

Signature

Date:

ACKNOWLEDGMENT

We would like to take this opportunity to express our sincere gratitude and thanks to all those who helped in the progress of the project entitled “wireless sensing networks”. Of all the persons who helped us, we would first of all like to thank **Prof. Rajeev kumar**, under whose able guidance we have worked on our project.

Date:

Name of the students:

Harshit Sharma(111081)

Sahil Sharma (111092)

Saurabh prabhakar (111136)

B.Tech (ECE)

Jaypee University of Information Technology

TABLE OF CONTENTS

List of Figures

ABSTRACT

Wireless Sensor Networks (WSNs) consist of small nodes with sensing, computation, and wireless communications capabilities. Many routing, power management, and data gathering protocols have been specially designed for WSNs where energy awareness is an essential design issue. The focus, however, has been given to the routing protocols which might differ depending on the application and network architecture. The clustering based hierarchical model imposes a structure on the network to achieve energy efficiency, stability, and scalability. In this model, network nodes are organized in clusters in which a node assumes the role of a cluster head. The cluster head is responsible for coordinating activities within the cluster and forwarding information between clusters. Clustering has potential to reduce energy consumption and extend the life time of the network. After it, we proposed PEGASIS (Power-Efficient Gathering in Sensor Information Systems) model, a near optimal chain-based protocol that is an improvement over clustering based model. In PEGASIS, each node communicates only with a close neighbour and takes turns transmitting to the base station, thus reducing the amount of energy spent per round.

Keywords: Wireless sensor network, data gathering cycle, Greedy Algorithm, static clustering (SC), network lifetime

CHAPTER 1

1.1 INTRODUCTION

The nodes of wireless sensor networks are defined with limited energy. Wireless sensor node deployed into the network to monitor the physical or environmental condition such as temperature, sound, vibration at different location. The data is transfer over the network each sensor consume some energy in receiving data, sending data. The lifetime of the network depend how much energy spent in each transmission. And how we can extend the lifetime of the nodes in which routers protocols play an important role. Wireless sensor network is a network that consists of hundreds of small nodes that are spatially distributed which monitor physical or environmental conditions, such as temperature, pressure, area monitoring, polluting monitoring, agriculture. Main concerns lies in transmission and since wireless communications consume significant amounts of battery power while transmitting and receiving data.

So we can classify energy consumption in WSN as useful and wasteful sources. The sensor networks are required to transmit gathered data to the base station (BS) or sink. Network lifetime thus becomes an important parameter for sensor network design as replenishing battery power of sensor nodes is an impractical proposition.

.

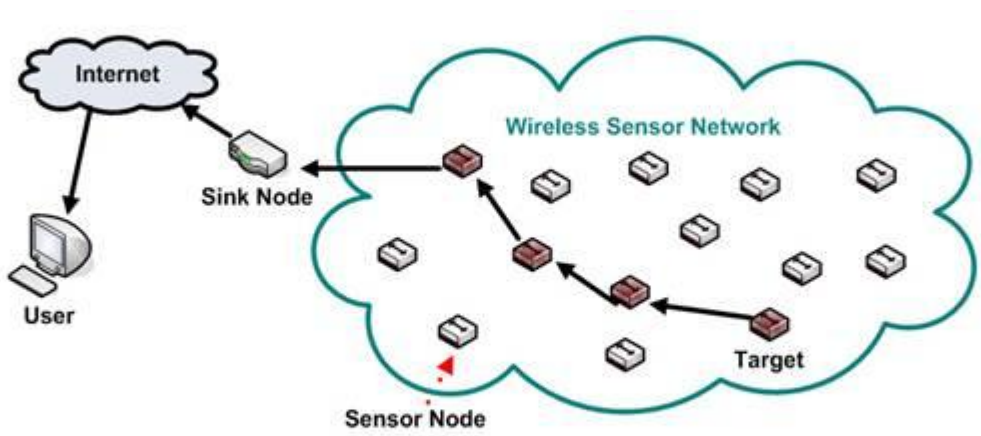

Figure 1

1.2 Structural view of Sensor Network

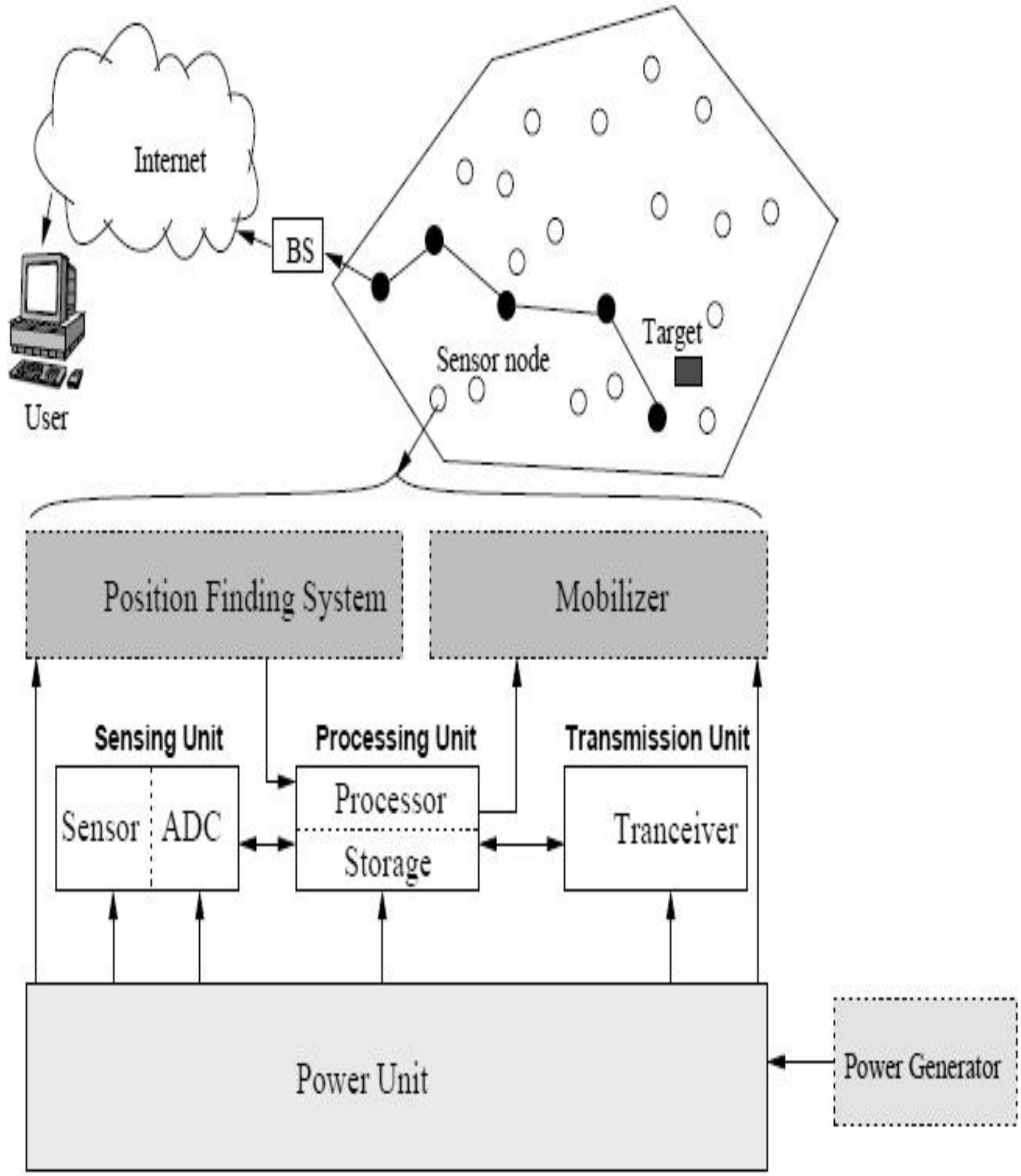

Figure 2

Basically each node consists of four basic components: sensor unit, central processing unit (CPU),power unit, and communication unit. Each component perform different task .The sensor unit consists of sensor and ADC requests, and returning the analogue data it sensed.ADC is a translator that tells the CPU what a sensor unit has sensed, and informs the sensor unit what to do. Communication unit is tasked to receive command or query from and transmit the data from CPU to the outside world.CPU interprets the command or query to ADC, monitors and controls power if necessary, processes received data, computes the next hop to the base station, etc. Power unit supplies power to sensor unit, processing unit and communication unit. Each node may also consist of the two optional namely Location finding system (GPS) and mobilzer if the user requires location with high accuracy.

Each node in a WSN consists of a small microprocessor, a power source (e.g., a battery or solar collector), a method of communication (such as an RF transmitter), and various sensors for monitoring the node's surroundings .Together, the nodes form an ad hoc network to monitor a physical space. When a node has a message to transmit (such as sample data or an alert message), the message is passed from node to node until it arrives at a central base station.

1.2.1 Classification of wireless sensor networks

Classification of sensor networks based on their mode of functioning and type of application.

Proactive Networks

The nodes in this type of network periodically, switch on their sensors and transmitters, sense the environment and transmit the data of interest. Hence, they provide the required parameters at regular intervals. They are well suited for application requiring periodic data monitoring.

Reactive Networks

The nodes of the networks according to this scheme react immediately to sudden and drastic changes in the value of a sensed parameter. They are well suited for time critical applications.

Hybrid Networks

The nodes in such a network not only react to time-critical situation, but also provide required parameters at regular intervals in a very energy efficient manner. In our heterogeneous network, we considered this hybrid form of deployment and functioning.

1.3 Routing Models

All known routing protocols may be included into one of the following three models.

1.3.1 one-hop model

This is the simplest approach and represents direct transmission. In these networks every node transmits to the base station directly. This approach implies not only to be too expensive in term of energy consumption, but it is also infeasible because nodes have limited transmission range.

Nodes in networks with large coverage usually are far enough thus their transmissions cannot reach the base station.

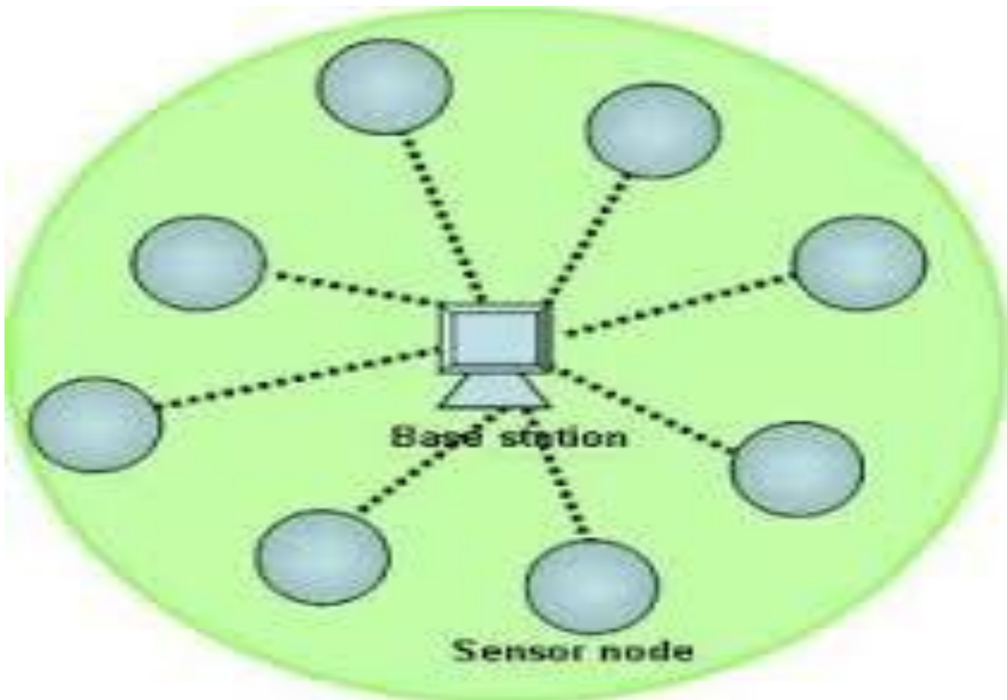

Figure 3

1.3.2 Multi-hop Planar Model

In this model, a node transmits to the base station by forwarding its data to one of its neighbours, which is closer to the base station .The latter passes on it to a neighbour that is even close to the base station. Therefore the information travels from source to destination by hop from one node to another until it reaches the destination.

A number of protocols employ this approach and some use other optimization techniques to enhance the efficiency of this model.

This network composed by thousands of sensors. This model will exhibit high data latency due to long time needed by the node information to arrive to the base station.

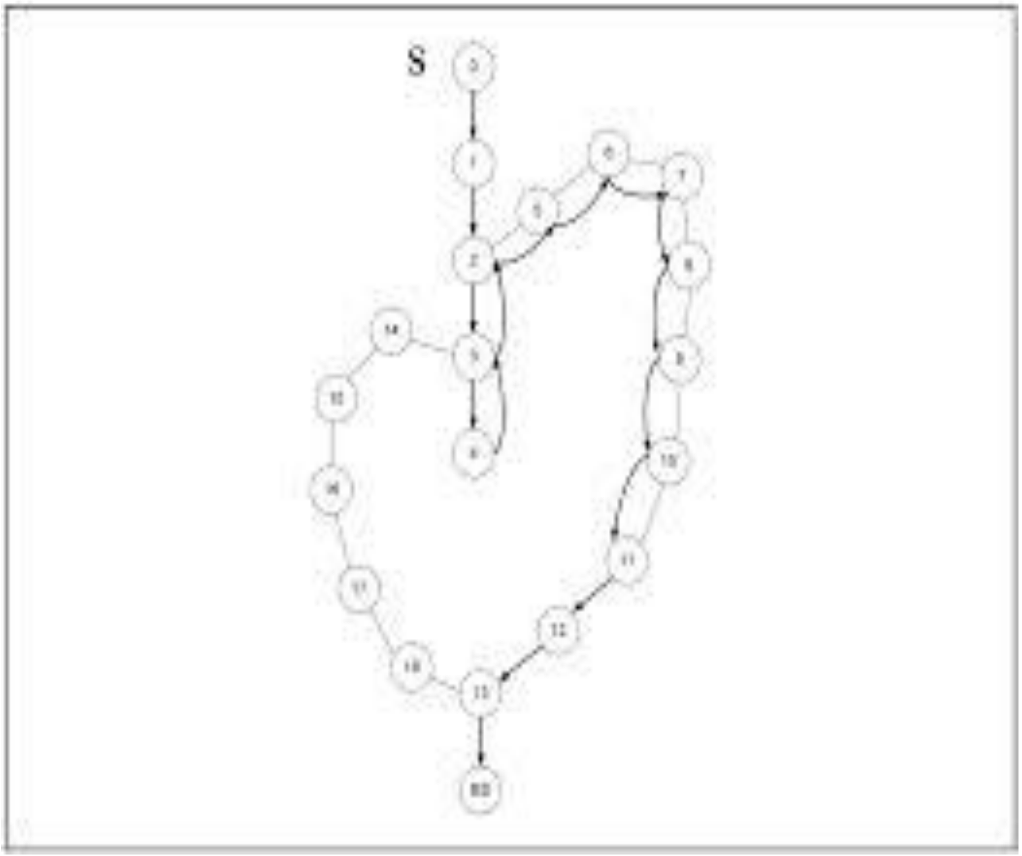

Figure 4

1.3.3 Clustering-based Hierarchical Model

This type of approach breaks the network topology into several areas called clusters. Nodes are grouped depending on some parameters into clusters with a cluster head, which has the responsibility of routing the data from the cluster to other cluster heads or base station. Data travels from a lower clustered layer to a higher one

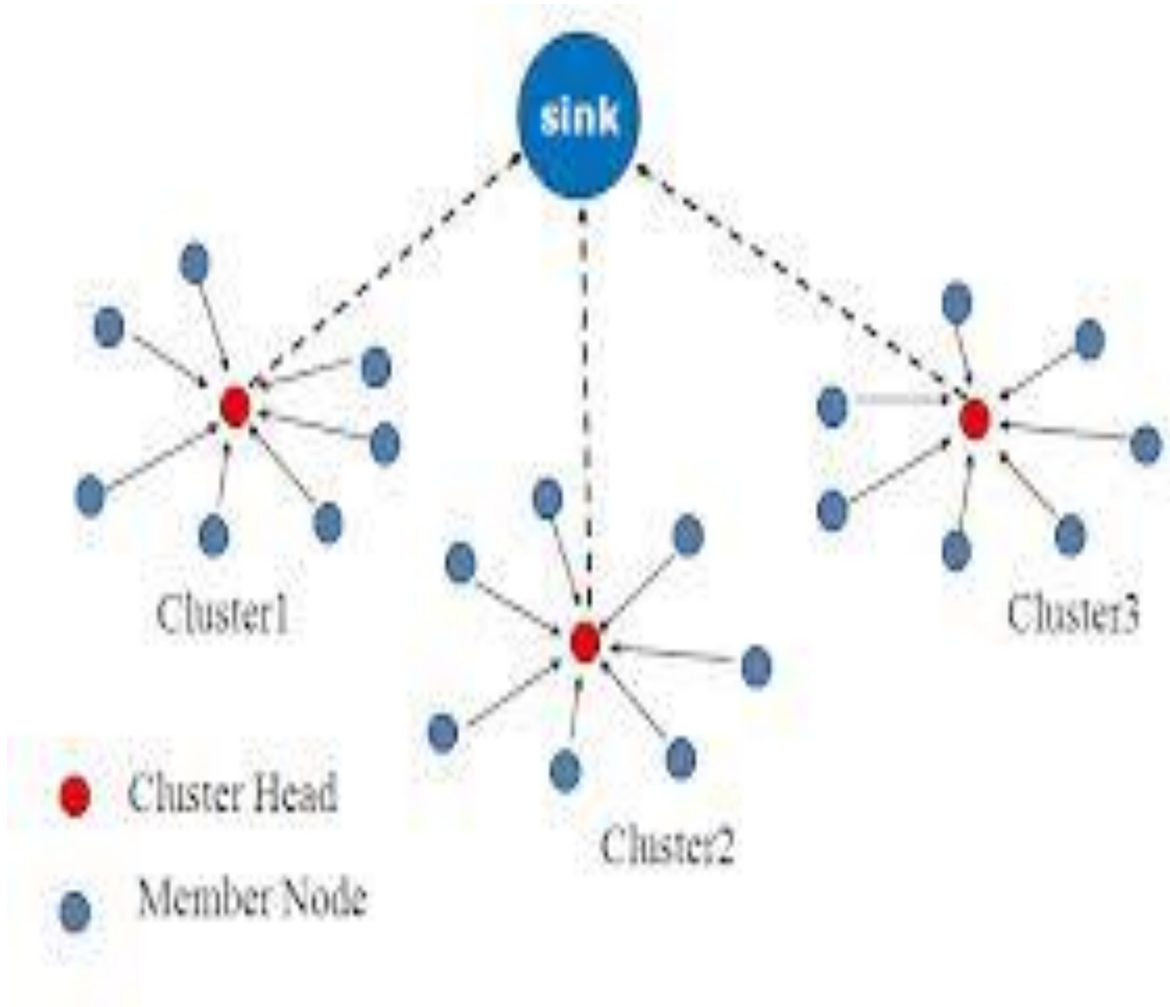

Figure 5

Data still hops from one nodes to another, but since it hops from one layer to another it covers large distances and moves the data faster to the base station than in the multi-hop model. The latency in this model is theoretically much less than in the multi-hop model. This model is more suitable than the one hop or multi-hop model.

1.4 DESIGN FACTORS

The design of routing protocols for WSN is challenging because of several network constrains. WSN suffer from limitations of several network resources, for example energy, bandwidth, control processing unit and storage. The design challenges in sensor networks involve the following main aspects.

LIMITED ENERGY CAPACITY

Since sensor nodes are battery powered, they have limited energy capacity. Energy poses a big challenge for network designers in different environment where it becomes difficult to charge the batteries. Furthermore, when the energy of a sensor reaches a certain threshold , the sensor will become faulty and will not be able to function properly, which will have a major impact on network performance.

SENSOR LOCATIONS

Another challenge that faces the design of routing protocols is to manage the locations of the sensors. Most of the routing protocols assume that the sensors are equipped with global positioning systems (GPS).

LIMITED HARDWARE RESOURCES

In addition to limited energy capacity, sensor nodes have also limited processing and storage capacities, and thus can only perform limited computational functionalities. These hardware constrains present may challenge in software development and network protocol design for sensor networks.

UNRELIABLE NETWORK

A sensor network usually operates in a dynamic and unreliable environment. The topology of a network, which is defined by the sensors and the communication links between the sensors, changes frequently due to sensor addition, deletion, node failures, damages, or energy depletion. Also, the sensor nodes are linked by a wireless network which is noisy and error prone.

QUALITY OF SERVICE

In some applications data must be transferred within a certain period of time from the moment it is sensed, or it will be useless. Therefore, bounded latency for data delivery is another condition for time constrained applications. As energy is depleted the network may be required to reduce the quality of service in order to reduce energy dissipation.

TRANSMISSION MEDIA

The required the bandwidth of sensor data will be low, on the order of 1-100kb/s. related to the transmission media is the design of MAC. One approach to MAC design is time division multiple access (TDMA) based protocols to conserve more energy.

1.5 HOMOGENEOUS AND HETEROGENOUS NETWORKS

1.5.1 HOMOGENEOUS NETWORK

A homogeneous sensor network consists of identical nodes. All the sensor nodes have same complexity, battery energy, and sensor range.

It is evident due to this the cluster head will be overloaded with the long range transmissions to the remote base station.

Due to which the cluster head expires before the other nodes. All the nodes will run out of battery at the same time causing problems in network.

Using a homogeneous network and role rotation, the nodes should have capability of acting as cluster heads, and possess the necessary hardware capabilities. With the advancement in the wireless sensor networks (WSN) various kinds of application specific protocols have been developed.

1.5.2 HETEROGENEOUS NETWORK

In this type of network all nodes have different functionality. There are basically two type of nodes 1 Cluster head node 2 sensing nodes

Sensing nodes sense required parameters and transmit it to cluster head. Cluster head gathers all the information from other nodes and send it to base station where all the computation takes place.

1.6 DESIGN FACTOR: ENERGY

Micro sensor data can contain hundreds or thousands of sensing nodes to sense data on the wireless links. So to fulfill this need these nodes are made as cheap and energy efficient as possible. These are deployed in large numbers to obtain high quality results.

Network protocols discussed must be fault tolerant and have minimum energy consumption.

In addition since the limited wireless channel bandwidth must be shared among all the sensors in the network, routing protocols for these networks should be able to perform local collaboration to reduce bandwidth requirements. Eventually the data being sensed by the nodes in the network must be transmitted to a control centre or base station where the end user can sense the data .There are many possible modes for these micro sensors networks .

In our work we consider micro sensor networks where the base station is fixed and located far from sensors

The node deployment is pre planned in which cluster head is at a fixed location from the base station and other sensors are deployed randomly around the cluster head.

We consider a heterogeneous network deployment model and range for cluster head is 100 meters and range for other sensors is 20 meters. The application of WSN we consider for our work is monitoring of the boundaries of long and wide field region.

The communication between the base station and sensor nodes is expensive, and there are no high energy nodes through which the communication can proceed. This is the framework for MITs-AMPS project, which focuses on

Innovative energy-optimized solutions at all levels of the system hierarchy, from the physical layers and communication protocols up to application layers and efficient dsp design for micro sensor nodes.

Sensor nodes contain too much data for an end user to process therefore an automated method of combining the data to meaningful information is required. In addition to avoid information overload, data aggregation

Also known as data fusion can combine several unreliable data measurements to produce a more accurate signal by enhancing the common signal and reduce the noise.

The classification performed on the aggregated might be performed by a human operator or automatically.

1.6.1 FIRST ORDER ENERGY MODEL

The use of clusters for transmitting data to the base station leverages the advantages of small distances for most nodes, requiring only a few nodes to transmit far distances to the base station.

In first order model, there is a great deal of research in the area of low energy radios. Different assumptions about the radio characteristics, including energy dissipation in the transmit and receive modes, will change the advantages of different protocols. In our work, we assume a simple model where the radio dissipates 50nj/bit to run the transmitter 100pj/bit/m_

For the transmit amplifier to achieve an acceptable.

These parameters are slightly better than the current state of the art in radio design. Receiving the message is not a low cost operation; the protocol must try to minimize not only the transmit distances but also the number of transmit and receive operations for each message

1.7 ALGORITHM

Steps-:

Step1-> BASE STATION POSITION IS DEFINED

Step2->POSITION OF FIRST CLUSTER IS DEFINED

Step3-> STRUCTURES S1 FOR OF FIRST CLUSTER IS
FUNCTION"RAND"

Step4->STRUCTURE S1 IS PLOTTED USING THE
FUNCTION "PLOT'

Step5->FOR STRUCTURE S1 ENERGY USED IN
1ROUND E1 AND THE NUMBER OF ROUNDS
N1 IS CALCULATED USING THE FUNCTION
“ENERGY”

Step6->POSITION OF SECOND CLUSTER HEAD IS
DEFINED

Step7-> STRUCTURES S2 FOR OF SECOND CLUSTER
IS FUNCTION”RANDDEPLOYMENT

Step8-> STRUCTURE S2 IS PLOTTED USING THE
FUNCTION “PLOTT”

Step9-> FOR STRUCTURE S2 ENERGY USED IN
1ROUND E2 AND THE NUMBER OF ROUNDS
N2 IS CALCULATED USING THE FUNCTION
“ENERGY”

.

.

.

.

Step10->POSITION OF LAST CLUSTER IS DEFINED

Step11-> STRUCTURES FOR OF LAST CLUSTER
IS MADE BY FUNCTION
”RANDDEPLOYMENT”

Step12-> LAST SUCTURE IS PLOTTED USING THE
FUNCTION “PLOTT”

Step13-> FOR THE LAST STRUCTURE ,ENERGY USED
IN 1 ROUND AND THE NUMBER OF
ROUNDS IT WILL UNDERGO IS
CALCULATED USING THE FUNCTION
“ENERGY”

Step14-> END PROGRAM

1.8 RESULT

BASE STATION

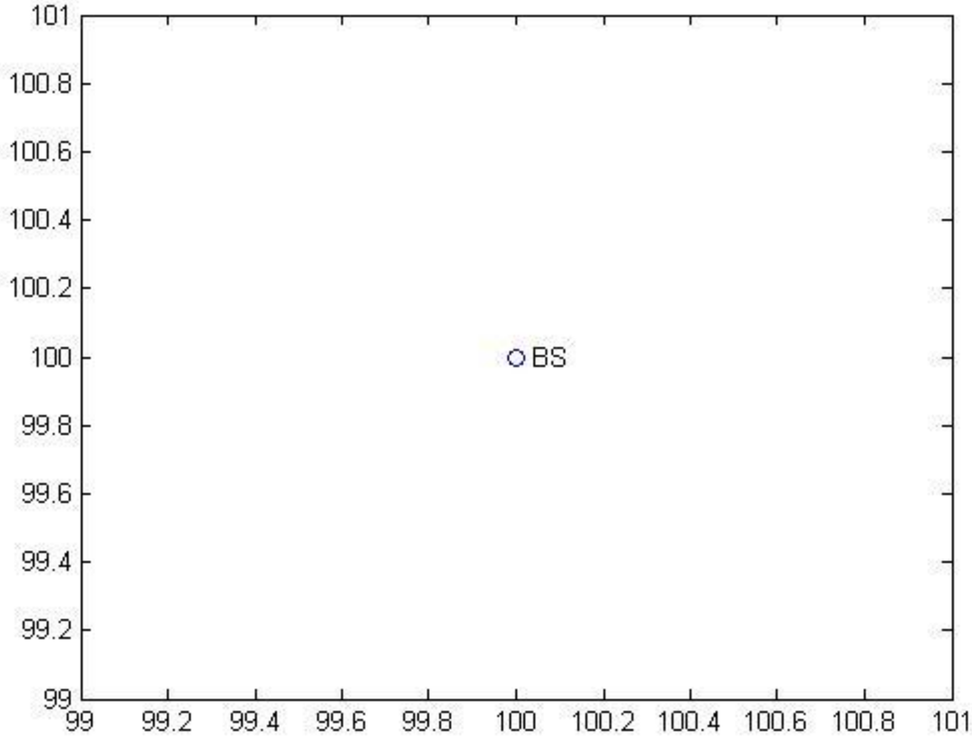

Figure 6

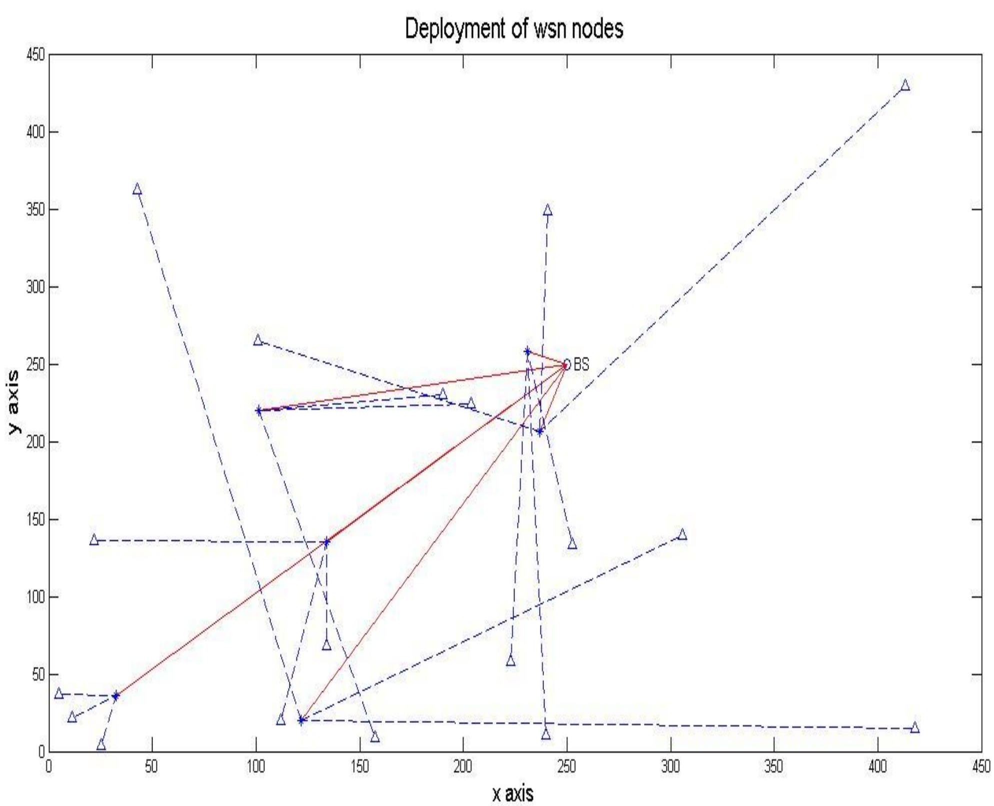

Figure 7

1.9 Applications:

1. **Military Application:**
   - **a.)** Monitoring friendly forces and equipment
   - **b.)** Military-theater or battlefield surveillance
   - **c.)** Targeting
   - **d.)** Battle damage assessment
   - **e.)** Nuclear, biological, and chemical attack detection and more . . .
2. **Environmental applications**
   - **a.)** Microclimates
   - **b.)** Forest fire detection
   - **c.)** Flood detection
   - **d.)** Precision agriculture and more . . .
3. **Health applications**
   - **a.)** Remote monitoring of physiological data
   - **b.)** Tracking and monitoring doctors and patients inside a hospital
   - **c.)** Drug administration
   - **d.)** Elderly assistance and more . . .
4. **Home applications**
   - **a.)** Home automation
   - **b.)** Instrumented environment

CHAPTER 2

2.1 PEGASIS: POWER EFFICIENT GATHERING IN SENSOR INFORMATION SYSTEMS

We present an improved protocol called PEGASIS (Power-Efficient Gathering in Sensor Information Systems), which is near optimal for data gathering application in sensor networks. The key idea in PEGASIS is to form a chain among the sensor nodes so that each node will receive from and transmit to a close neighbor. Gathered data moves from node to node, get fused, and eventually a designated node transmits to the BS. Nodes take turns transmitting to the BS so that the average energy spent by each node per round is reduced. Building a chain to minimize the total length is similar to the traveling salesman problem, which is known to be intractable. However, with the radio communication energy parameters, a simple chain built with a greedy approach performs quite well. To ensure balanced energy dissipation in the network, an additional parameter could be considered to compensate for nodes that must do more work every round. If the sensor nodes have different initial energy levels, then we could consider the remaining energy level for each node in addition to the energy cost of the transmissions. The assumption of location information is not critical. The BS can determine the locations and transmit to all nodes, or the nodes can determine this through received signal strengths. For example, nodes could transmit progressively reduced signal strengths to find a close neighbor to exchange data. This would require the nodes to consume some energy when trying to find local neighbours, however, this is only a fixed initial energy cost when constructing the chain. If nodes are mobile, then different methods of transmission could be examined. For instance, if nodes could approximate how often and at what speed other nodes are moving, then it could determine more intelligently how much power is needed to reach the other nodes. Perhaps, the BS can help coordinate the activities of nodes in data transmissions.

Main concerns lie in transmission and since wireless communications consume significant amounts of battery power while transmitting and receiving data. So we can classify energy consumption in WSN as useful and wasteful sources. Useful energy consumption can be due to

1. Transmitting/receiving data,
2. Processing query requests,
3. Forwarding queries/data to neighboring nodes.

Wasteful energy consumption can be due to

1. Idle listening to the media,
2. Retransmitting due to packet collisions.

Therefore we should optimize them so that little energy is spent while receiving and transmitting data. Initially protocols are not adapted according to the requirement so we proposed PEGASIS techniques. It can aid in reducing useful energy consumption.

The main idea in PEGASIS is for each node to receive from and transmit to close neighbors and take turns being the leader for transmission to the BS. This approach will distribute the energy load evenly among the sensor nodes in the network. We initially place the nodes randomly in the play field, and therefore, the *i*–th node is at a random location. The nodes will be organized to form a chain, which can either be accomplished by the sensor nodes themselves using a greedy Algorithm starting from some node. Alternatively, the BS can compute this chain and broadcast it to all the sensor nodes.

For constructing the chain, we assume that all nodes have global knowledge of the network and employ the greedy algorithm. We could have constructed a loop, however, to ensure that all nodes have close neighbors is difficult as this problem is similar to the traveling salesman problem. The greedy approach to constructing the chain works well and this is done before the first round of communication. To construct the chain, we start with the furthest node from the BS. We begin with this node in order to make sure that nodes farther from the BS have close neighbors, as in the greedy algorithm the neighbor distances will increase gradually since nodes already on the chain cannot be revisited.

2.2 GENERAL PEGASIS MODEL

The PEGASIS (Power-Efficient Gathering in Sensor Information Systems) protocol forms a chain of the sensor nodes and the chain is formed using a greedy approach, starting from the node farthest to the sink node. The nearest node is sending the data to the neighbour node. This procedure is continued until all the nodes are included in the chain. This approach will distribute the energy load evenly among the sensor nodes in the network. Here before passing the information to the adjacent neighbour data aggregation takes place.

Clusters. In PEGASIS all nodes communicates with their closest neighbours and continues their communication until the aggregated data reaches the BS. Thus this improves the network lifetime; since it reduces the power consumption required per round. PEGASIS protocol also consist of rounds. They are:

- Chain Formation.
- Leader Selection.
- Data Transmission.

CHAIN FORMATION:

For constructing the chain the PEGASIS protocol starts from furthest from the base station and uses Greedy algorithm to form chain. Node C0 lies furthest from base station. So chain starts from C0.After that node C1 is selected and so on till node C5.

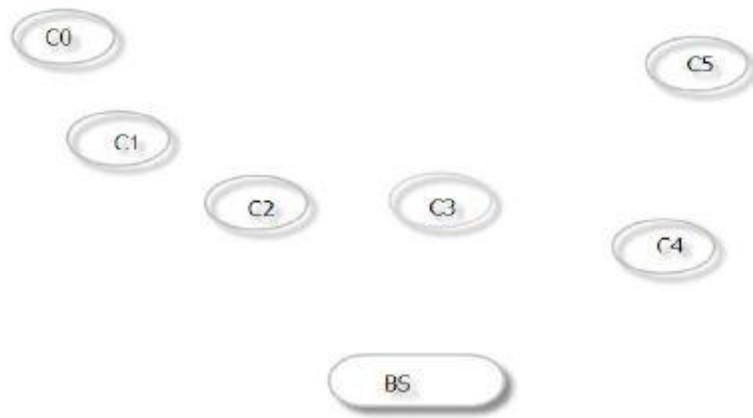

Figure 8

GREEDY ALGORITHM

A **greedy algorithm** is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

For example, a greedy strategy for the travelling salesman problem (which is of a high computational complexity) is the following heuristic: "At each stage visit an unvisited city nearest to the current city". This heuristic need not find a best solution, but terminates in a reasonable number of steps; finding an optimal solution typically requires unreasonably many steps

A greedy algorithm is a mathematical process that looks for simple, easy-to-implement solutions to complex, multi-step problems by deciding which next step will provide the most obvious benefit.

Such algorithms are called greedy because while the optimal solution to each smaller instance will provide an immediate output, the algorithm doesn’t consider the larger problem as a whole. Once a decision has been made, it is never reconsidered.

Leader Selection

At the beginning of each round leader node is selected randomly. The beneath of selecting the random node is that if the node dies at random location the network will be robust. After the leader is selected a token is passed to the end node to initiate the data gathering. Passing token also consumes energy but the size of token is so small that the cost for passing is very small.

Data Transmission

The node who has the token starts sending its data to its neighbour. The neighbour node fuses its data with the data which it has received and then passes the data to its neighbour. This process continues till it reaches the leader node .Then the leader node transmit the fused data to the BS.Thus leader node is then rotated randomly in each round.

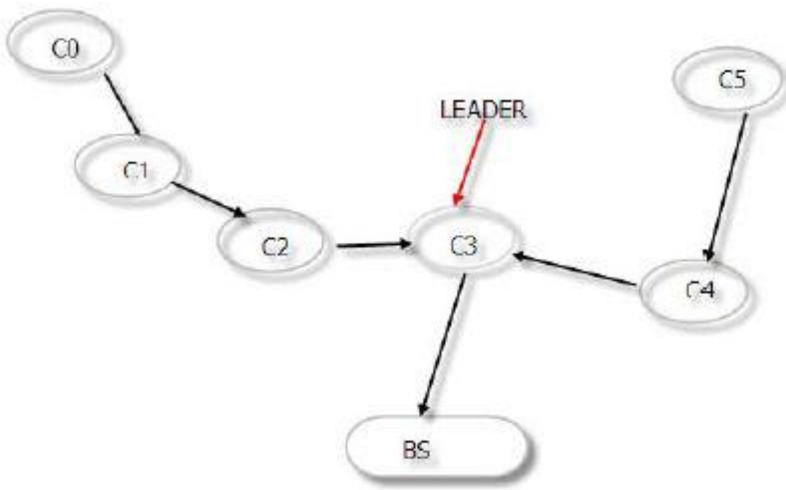

Figure 9

Gathering data

In this step after leader node is selected and chain is constructed next step is data aggregation and leader node is responsible for forwarding the aggregated data to the sink node.

2.3 First order radio model

We consider the first order radio model with identical parameter values. The energy spent
in transmission of a single bit is given by

etx(d) = et1 + ed1*d^n

Where et1 is the energy dissipated per bit in the transmitter circuitry and ed1*d^n is the energy dissipated for transmission of a single bit over a distance d, n being the path loss exponent (usually $2.0 \leq n \leq 4.0$). For simulation purposes we have considered a first order model where we assume n=2. This is the value of n obtained for free space. As channel non-linearities increase the value of n enhances. An increase in the value of n would help our model would gain even greater relevance as BS transmission would then require greater energy dissemination. Thus the total energy dissipated for transmitting a K-bit packet is

ETx (k, d) = ETx– elec (k) + ETx–amp (k,d)

ETx(K,d) = (et1 + ed1*d^2) * K

= et + ed*d^2

where **et =et1*K** and **ed= ed1*K**

If er1 be the energy required per bit for successful reception then the energy dissipated for receiving a K-bit packet is

Receiving

ERx(k) = ERx-elec(k)

ERx(K) = er1 * K

= er

where **er =er1*K**

Receiving is also a high cost operation, therefore, the number of receives and transmissions should be minimal.

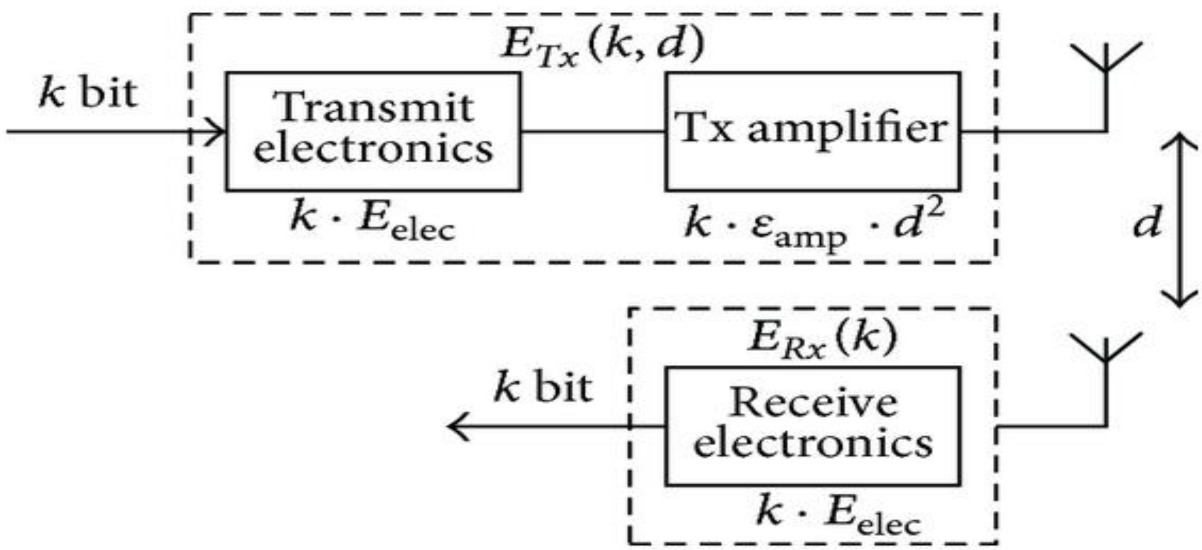

Figure 10

receives and transmissions should be minimal. LEACH and PEGASIS use the same constants (Eelec, ed1, and k) for calculating energy costs, therefore the PEGASIS achieves its energy savings by minimizing d and the number of transmissions and receives for each node. Therefore, for a d^4 model, PEGASIS would achieve even greater savings compared to LEACH.

2.4 Energy Cost Analysis for Data Gathering

In this section we will analyze the cost of data gathering from sensor nodes to the distant BS. The data collection problem of interest is to send a k-bit packet from each sensor node in each round. Of course, the goal is to keep the sensor nodes operating as long as possible. A fixed amount of energy is spent in receiving and transmitting a packet in the electronics, and an additional amount proportional to d^2 is spent while transmitting a packet.

With the direct approach, all nodes transmit directly to the BS which is usually located very far away. Therefore, every node will consume a significant amount of power to transmit to the BS in each round. Since the nodes have a limited amount of energy, nodes will die quickly, causing the

reduction of the system lifetime. As the direct approach would work best if the BS is located close to the sensor nodes or the cost of receiving is very high compared to the cost of transmitting data.

For the rest of the analysis, we assume a 100-node sensor network with the BS located far away. In this scenario, energy costs can be reduced if the data is gathered locally among the sensor nodes and only a few nodes transmit the fused data to the BS. This is the approach taken in LEACH, where clusters are formed dynamically in each round and cluster-heads (leaders for each cluster) gather data locally and then transmit to the BS. Cluster-heads are chosen randomly, but all nodes have a chance to become a cluster-head in LEACH, to balance the energy spent per round by each sensor node

Although this approach is about 8x better than the direct transmission, there is still some room to save even more energy. The cost of the overhead to form the clusters is expensive. In LEACH, in every round 5% of nodes are cluster-heads, and they must broadcast a signal to reach all nodes. In addition, several cluster-heads transmit the fused data from the cluster to the distant BS. Further improvement in energy cost for data gathering can be achieved if only one node transmits to the BS per round and if each node transmits only to local neighbours in the data fusion phase. This is done in the PEGASIS protocol to obtain an additional factor of 2 or more improvement compared to LEACH. The energy spent in each node for 100 rounds is about 100*.0002 Joules for the electronics and at least .002 Joules for one message transmission to the

BS. With an initial energy in each node to be .25 Joules, the maximum number of rounds possible before a node dies is approximately 1100. The actual number will be less since we did not account for the energy spent in a node for local transmission, which depends on distance, and the cost for data fusion. Therefore, the upper bound will likely be less than 1000 rounds. The PEGASIS protocol achieves about 800 rounds, which is near optimal.

2.5 ALGORITHM:

It is possible that some nodes may have relatively distant neighbour nodes along the chain in PEGASIS. On one hand, nodes already on the chain cannot be revisited. On the other hand, when a node dies, the chain is reconstructed in the same manner (greedy algorithm) to bypass the dead node.

The algorithm uses the following steps to form a chain:

1) Initialize the network parameters. Determine the number of nodes, BS location information et al. Then chain construction starts.
2) BS broadcasts the whole network a message to obtain basic network information such as ID of nodes alive and distance from each node to BS.
3) Set the node which is farthest from BS as end node, it joins the chain first and is labelled as node 1.
4) End node of the chain obtains the information of distance between itself and other nodes which have not joined the chain yet, finds the nearest node and sets it as node I waiting to join the chain, i represents the i-th node joined.

2.6 RESULT:

PEGASIS MODEL

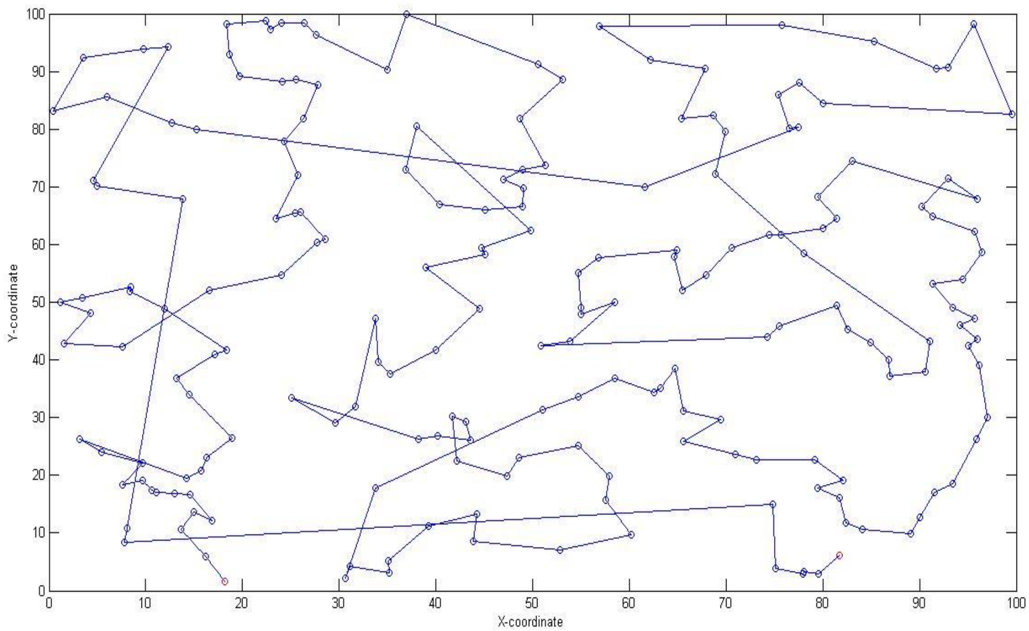

Figure 11

2.7 CONCLUSION:

In this we describe PEGASIS; it is chain based protocol that is near optimal for a data-gathering problem in sensor networks. PEGASIS outperforms LEACH by eliminating the overhead of dynamic cluster formation, minimising the distance non leader-nodes must transmit, limiting the number of transmissions and receives among all nodes, and using only one transmission to the BS per round. The proposed work is implemented on Wireless Sensor network to improve the network life in case of chain based protocol.

In this we propose energy efficient PEGASIS routing scheme to increase network lifetime of sensor networks. Our scheme achieves balance of energy dissipation among the nodes and to increase the existence of more nodes in the network. We evaluate performance of our scheme by using simulation on matlab.

APPENDIX

Matlab code:

```
clear;
clc;
clf;

NodeNums=200;

AreaR=100;

Bx=50;
By=175;
Tr=100;
%for NodeNums=100:20:400

num=1;
num_plot=1;
En=0.25;
send_to_sink=0;
ctl_pkt_leng=100;
data_pkt_length=2000;
die_node_num=0;
die_node_num_pri=0;
run_round=0;
transmited_packet=0;
En_Cost=0;
En_Cost_pre=0;
inter_cost=0;
```

```
begin_to_send=0;
En_cost_per_round=0;


alive=1;
dead=0;


Node.x=AreaR*rand(1,NodeNums); % the position of node
Node.y=AreaR*rand(1,NodeNums);
%Node.x(100)=AreaR/2;
%Node.y(100)=AreaR/2;
Node.pri=linspace(0,0,NodeNums);
Node.already=linspace(0,0,NodeNums);
Node.to_nbr_dis=zeros(NodeNums);
Node.to_pri_dis=linspace(0,0,NodeNums);
Node.send_dis=linspace(0,0,NodeNums);
Node.E=linspace(En,En,NodeNums);
Node.status=linspace(alive,alive,NodeNums);
Node.E_dis=linspace(0,0,NodeNums);


%Eelec=Etx=Erx
ETX=50*0.000000001;
ERX=50*0.000000001;
%Transmit Amplifier types
Efs=100*0.000000000001;
Emp=0.0013*0.000000000001;
%Data Aggregation Energy
EDA=5*0.000000001;
do=sqrt(Efs/Emp);
```

```
for i=1:1:NodeNums
        To_sink_dist(i)=sqrt((Node.x(i)-Bx)^2+(Node.y(i)-By)^2);
end

%[Max_Dis, Max_num]=max(to_sink_dist);
%now_node=Max_num;
%Node.already(now_node)=1;
%Max_num
%now_node
%node_head=Max_num;
%send_to_sink=now_node;
%node_to_send=now_node;
%start_node=now_node;

%for nodes=1:1:100

%Node.pri=linspace(0,0,NodeNums);
%Node.already=linspace(0,0,NodeNums);
%Node.to_nbr_dis=zeros(NodeNums);
%Node.to_pri_dis=linspace(0,0,NodeNums);
%Node.send_dis=linspace(0,0,NodeNums);
%Node.E=linspace(En,En,NodeNums);
%Node.status=linspace(alive,alive,NodeNums);
%Node.E_dis=linspace(0,0,NodeNums);
%run_round=0;
%send_to_sink=0;
%die_node_num=0;
```

```
%die_node_num_pri=0;

%begin_to_send=0;


while run_round<1 %die_node_num<1


%        num_plot=1;

%                        die_node_num_pri=die_node_num;

         En_Cost_pre=En_Cost;

%        run_round

%        die_node_num

%        En_Cost


                  for i=1:1:NodeNums

                            if Node.status(i)==alive

%                                     Node.E_dis(i)=Node.E(i)/(ETX+Efs*(To_sink_dist(i)^2));

                                      Node.E_dis(i)=Node.E(i);

                            else

                                      Node.E_dis(i)=0;

                            end

                  end


         if run_round==0||die_node_num_pri~=die_node_num


                  Node.pri=linspace(0,0,NodeNums);

                  num_plot=1;

                  Node.already=linspace(0,0,NodeNums);

%                 run_round

                  for i=1:1:NodeNums
```

```
            if Node.status(i)==alive
                to_sink_dist(i)=sqrt((Node.x(i)-Bx)^2+(Node.y(i)-By)^2);
            else
                to_sink_dist(i)=0;
            end
        end

        [Max_Dis, Max_num]=max(to_sink_dist);
        now_node=Max_num;
        Node.already(now_node)=1;
        node_head=Max_num;
%       send_to_sink=now_node;
        node_to_send=now_node;
        start_node=now_node;
%       now_node


        while num_plot~=NodeNums-die_node_num
                    if num_plot==NodeNums-die_node_num
                        Node.pri(now_node)=0;
                        node_tail=now_node;
                    end



                    for j=1:1:NodeNums
                            if Node.already(j)==0&&Node.status(j)==alive

Node.to_nbr_dis(now_node,j)=sqrt((Node.x(now_node)-Node.x(j))^2+(Node.y(now_node)-Node.y(j))^2);
                            else
```

```
                    Node.to_nbr_dis(now_node,j)=0;
                end
            end

            j=1;
            while Node.to_nbr_dis(now_node,j)==0
                    j=j+1;
            end

            min_dis=Node.to_nbr_dis(now_node,j);
            min_num=j;
            for j=1:1:NodeNums
                if
Node.to_nbr_dis(now_node,j)~=0&&Node.to_nbr_dis(now_node,j)<min_dis
                    min_dis=Node.to_nbr_dis(now_node,j);
                    min_num=j;
                end
            end
            Node.to_pri_dis(now_node)=min_dis;
            Node.pri(now_node)=min_num;

            Node.E(now_node)=Node.E(now_node)-
(ETX*ctl_pkt_leng+Efs*2*(min_dis* min_dis)*ctl_pkt_leng);
            Node.E(Node.pri(now_node))=Node.E(Node.pri(now_node))-
ETX*ctl_pkt_leng;

            if now_node==Max_num
                Node.to_pri_dis(now_node);
            end
            now_node=Node.pri(now_node);
```

```
%                               min_num
                                Node.already(now_node)=1;
%                               Node.to_pri_dis(now_node)=min_dis;
%                               min_dis
                                num_plot=num_plot+1;
            end

            node_tail=now_node;



            I_want=1;
        if I_want==1
            figure(1);
            if run_round~=1
                clf;
            end
%           plot(Bx,By,'+');
%           hold on;

            for i=1:1:NodeNums
                if Node.status(i)==alive
                    plot(Node.x(i), Node.y(i), 'o','markersize',5);
                    hold on;
                end
            end

            plot(Node.x(node_tail), Node.y(node_tail), 'o-r','markersize',5);
```

```
%               plot(Node.x(node_head), Node.y(node_head), 'o-r','markersize',5);

                for i=1:1:NodeNums
                        if Node.status(i)==alive
                                if Node.pri(i)~=0
                                        plot([Node.x(i);Node.x(Node.pri(i))],[Node.y(i);Node.y(Node.pri(i))]);
                                        hold on;
                                end
                        else
                                        plot(Node.x(i), Node.y(i), 'o-r');
                        end
                end
        end
        xlabel('X-coordinate');
        ylabel('Y-coordinate');

%
        plot([Node.x(send_to_sink);Bx],[Node.y(send_to_sink);By]);
%                                       hold on;
%               node_to_send=Max_num;
%               start_node=Max_num;

        end

%       send_to_sink=node_head;
%       next_next=mod(run_round,NodeNums-die_node_num);
%       while next_next~=0
%                               send_to_sink=Node.pri(send_to_sink);
%                               next_next=next_next-1;
```

```
%       end
%               [max_node_En,send_to_sink]=max(Node.E_dis);
                send_to_sink=mod(begin_to_send,NodeNums)+1;
                run_round=run_round+1;
                send=send_to_sink;
                while Node.status(send_to_sink)~=alive
                                send_to_sink=mod(send_to_sink,NodeNums)+1;
                end
                begin_to_send=send_to_sink+1;
                transmited_packet=transmited_packet+NodeNums-die_node_num;

        die_node_num_pri=die_node_num;
%       for i=1:1:NodeNums
%                       if Node.E(i)<0.01*En
%                               die_node_num_pri=die_node_num_pri+1;
%                        end
%       end

        num=1;
        right_send_num=1;
        node_to_send=node_head;
        start_node=node_head;

                while node_to_send~=send_to_sink
                                right_send_num=right_send_num+1;
                                node_to_send=Node.pri(node_to_send);
                end
%               right_send_num
```

```
		Node.send_dis=linspace(0,0,NodeNums);
		while num~=NodeNums+1-die_node_num
				if num<right_send_num
					Node.send_dis(start_node)=Node.to_pri_dis(start_node);
					start_node=Node.pri(start_node);
				end
				if num==right_send_num
					Node.send_dis(start_node)=to_sink_dist(start_node);
					start_node_1=start_node;
					start_node=Node.pri(start_node);
				end
				if num>right_send_num
					Node.send_dis(start_node)=Node.to_pri_dis(start_node_1);
					start_node_1=start_node;
					start_node=Node.pri(start_node);
				end
				num=num+1;
		end

		inter_cost=0;
		for i=1:1:NodeNums
			if i~=send_to_sink && Node.status(i)==alive
				inter_cost=inter_cost+ETX+ERX+Efs*Node.send_dis(i).^2;
			end
		end
%		inter_cost/(NodeNums-die_node_num)

	if send_to_sink==node_head
```

```
            for i=1:1:NodeNums
                if Node.status(i)==alive
                    if i==send_to_sink
                            if Node.send_dis(i)>do
%                               Node.E(i)=Node.E(i)-(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Emp*(Node.send_dis(i).^4)*data_pkt_length);
                                Node.E(i)=Node.E(i)-(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);
                            else
                                Node.E(i)=Node.E(i)-(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);
                            end
                    end
                    if i==node_tail
                        if Node.send_dis(i)>do
%                               Node.E(i)=Node.E(i)-(ETX*data_pkt_length+Emp*(Node.send_dis(i).^4)*data_pkt_length);
                                Node.E(i)=Node.E(i)-(ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);
                            else
                                Node.E(i)=Node.E(i)-(ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);
                            end
                    end
                    if i~=send_to_sink&&i~=node_head&&i~=node_tail
                            if Node.send_dis(i)>do
%                               Node.E(i)=Node.E(i)-(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Emp*(Node.send_dis(i).^4)*data_pkt_length);
```

```
                                        Node.E(i)=Node.E(i)-
(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_
length);
                                    else
                                        Node.E(i)=Node.E(i)-
(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_
length);
                                    end
                          end
                     end
                end
           end


           if send_to_sink==node_tail
                for i=1:1:NodeNums
                     if Node.status(i)==alive
                          if i==send_to_sink
                                    if Node.send_dis(i)>do
%                                       Node.E(i)=Node.E(i)-
(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Emp*(Node.send_dis(i).^4)*data_pkt
_length);
                                        Node.E(i)=Node.E(i)-
(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_
length);
                                    else
                                        Node.E(i)=Node.E(i)-
(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_
length);
                                    end
                          end
                          if i==node_head
                               if Node.send_dis(i)>do
```

```
%                                                       Node.E(i)=Node.E(i)-
(ETX*data_pkt_length+Emp*(Node.send_dis(i).^4)*data_pkt_length);

                                                        Node.E(i)=Node.E(i)-
(ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);

                                                else

                                                        Node.E(i)=Node.E(i)-
(ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);

                                                end

                                        end

                                        if i~=send_to_sink&&i~=node_head&&i~=node_tail

                                                if Node.send_dis(i)>do

%                                                       Node.E(i)=Node.E(i)-
(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Emp*(Node.send_dis(i).^4)*data_pkt
_length);

                                                        Node.E(i)=Node.E(i)-
(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_
length);

                                                else

                                                        Node.E(i)=Node.E(i)-
(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_
length);

                                                end

                                        end

                                end

                        end

                end


                if send_to_sink~=node_tail&&send_to_sink~=node_head

                        for i=1:1:NodeNums

                                if Node.status(i)==alive

                                        if i==send_to_sink

                                                if Node.send_dis(i)>do
```

```
%                                        Node.E(i)=Node.E(i)-(2*ERX*data_pkt_length+EDA*3*data_pkt_length+ETX*data_pkt_length+Emp*(Node.send_dis(i).^4)*data_pkt_length);

                                         Node.E(i)=Node.E(i)-(2*ERX*data_pkt_length+EDA*3*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);

                                   else

                                         Node.E(i)=Node.E(i)-(2*ERX*data_pkt_length+EDA*3*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);

                                   end

                      end

                      if i==node_head||i==node_tail

                            if Node.send_dis(i)>do

%                                        Node.E(i)=Node.E(i)-(ETX*data_pkt_length+Emp*(Node.send_dis(i).^4)*data_pkt_length);

                                         Node.E(i)=Node.E(i)-(ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);

                                   else

                                         Node.E(i)=Node.E(i)-(ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);

                                   end

                      end

                      if i~=send_to_sink&&i~=node_head&&i~=node_tail

                                   if Node.send_dis(i)>do

%                                        Node.E(i)=Node.E(i)-(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Emp*(Node.send_dis(i).^4)*data_pkt_length);

                                         Node.E(i)=Node.E(i)-(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);

                                   else
```

```
                                        Node.E(i)=Node.E(i)-(ERX*data_pkt_length+EDA*2*data_pkt_length+ETX*data_pkt_length+Efs*(Node.send_dis(i).^2)*data_pkt_length);
                                    end
                          end
                     end
                end
           end

%          for k=1:1:NodeNums
%                    if Node.pri(k)~=0
%                        En_cost_per_round=En_cost_per_round+(ERX+ETX)*data_pkt_length+Efs*data_pkt_length*(Node.send_dis(k).^2);
%                    else
%                        En_cost_per_round=En_cost_per_round+(ERX+ETX)*data_pkt_length+Efs*data_pkt_length*(Node.send_dis(k).^2);
%                    end
%          end
     En_cost_per_round=0.25*100-sum(Node.E);
     En_Cost=En*NodeNums-sum(Node.E);
     round_cost=En_Cost-En_Cost_pre;
%    round_cost
%    send_to_sink
%    Node.E
%    if Node.pri(send_to_sink)==0
%              send_to_sink=node_head;
%    else
%              send_to_sink=Node.pri(send_to_sink);
%    end
```

```
        die_node_num=0;
        for i=1:1:NodeNums
                if Node.E(i)<0.001*En
                        die_node_num=die_node_num+1;
                        Node.status(i)=dead;
                end
        end
end
run_round
%plot(NodeNums,run_round,'o-r');
%hold on;
%end
%plot(run_round,NodeNums-nodes,'o-r');
%hold on;
%end
```

REFERENCES

[1] I. Mohammad, and M. Imad, Handbook of sensor networks, CRC Press, London, 2005

[2] Energy efficient communication protocol for wireless microsensor networks, Wendi Rabiner Heinzelman, Anantha Chandrakasan & Hari Balakrishnan; Massachusetts Inst. Of tech. Cambridge, MA 02139

[3] A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks", proc. 33rd hawai inst conf. Sys. Sci. (HICSS'00), 2000.

[4] Proceedings of the First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06) 0-7695-2614-4/06 $20.00 © 2006 **IEEE**

[5] Wendi Heinzelman, Anantha Chandrakasan and Hari Balakrishna,"Energy-Efficient Communication Protocol for Wireless MicrosensorNetworks", In Proceedings of 33rd Hawaii International Conference on System Sciences, Jan. 2000, pp. 1-10;

[6] S. Lindsey, C. S. Raghavendra, "PEGASIS: Power Efficient Gathering in Sensor Information Systems", In Proceedings of IEEE ICC 2001, pp. 1125-1130, June 2001;

[7] D. Estrin, R.Govindan, J.Hiedemann and Satish Kumar, "Next Century Challenges: Scalable Coordination in sensor networks", In Proceedings of Mobicom '99, 1999;

[8] Yunxia Chen and Qing Zhao, "On the Lifetime of Wireless Sensor Networks", IEEE Communications Letters, Vol. 9, No. 11, November 2005;

[9] S.Lindsey, C.S.Raghavendra and K.Sivalingam, "Data Gathering in Sensor Networks using energy*delay metric", In Proceedings of the 15th International Parallel and Distributed Processing Symposium 2001, pp.

[10] Wendi Heinzelman, J.Kulik and H.Balakrishna, "Adaptive Protocols for information dissemination in Wireless Sensor Networks", *In Proceedings of ACM MobiCom '99*, pp.174-85;

[11] F.Ye, A.Chan, S.Lu and L.Zhang, "A Scalable Solution to Minimum Cost Forwarding in Large Sensor Networks", In *Proceedings of IEEE ICCCN 2001*, pp. 304-309, October 2001;

[12] Chandrakasan, Amirtharaja, Cho, Goodman, Konduri,Kulik, Rabiner and Wang, "*Design Considerations for Distributed Microsensor Systems", In IEEE 1999 Custom Integrated Circuit Conference (CICC)*, pp. 279-286, May 1999;

[13] Marco Dorigo **"***Ant Algorithms Solve Difficult Optimization Problems*", *In Proceedings of the 6th European Conference on Advanced Artificial Life*, pp 11- 22, 2001;

[14] Jun Ouyang**,** Gui_Rong Yan "*A Multi-Group Ant Colony System Algorithm For TSP*", *In proceedings of the Third International Conference on Machine Learning and Cyberncdcs, Shanghai, 26-29 August 2004.*

[15] K. M. Sivalingam, M. B. Srivastava and P. Agrawal. Low Power Link and Access Protocols for Wireless Multimedia Networks. In *Proceedings IEEE Vehicular Technology Conference VTC'97*, May 1997.

[16] R. Steele. *Mobile Radio Communications*. Pentech Press, London, 1992.

[17] M. Stemm, P. Gauthier, D. Harada and R. Katz. Reducing Power Consumption of Network Interfaces in Hand-Held Devices. In *Proceedings 3rd Intl. Workshop on Mobile Multimedia Communications*, Sept. 25-27, 1996, Princeton, NJ.

[18] The WINS Project. http://www.janet.ucla.edu/WINS.

[19] M. Zorzi and R. R. Rao. Energy Management in Wireless Communications. In *Proceedings 6th WINLAB Workshop on Third Generation Wireless Information Networks*, March 1997.

BIBLIOGRAPHY:

1. International journal of computer applications (0975-8887) volume 27-no.4, August 2011

2. www.mdpi.com/journal/algorithms 6,2013,29-42

3. Routing techniques in wireless sensor networks: a survey, Jamal N. Al-Karaki, Ahmed E. Kamal, Deptt. of Electrical & Computer engg.; Lowa state university, Ames.

4. Wireless sensor network: a survey, I.F. Akyildiz, W.Su, Y. Sankarasubramaniam & E. Cayirci, 20 December 2001

5. Wireless sensor networks: technology, protocols and applications; Kazem Sohraby, Daniel Minoli, Taieb Znati.

6. www.google.com

7. www.wikipedia.com